

Питання на екзамен

- 1. Архітектура системи лінгвістичного аналізу текстів**
- 2. Редакційна відстань.**
- 3. Методи виправлення орфографічних помилок.**
- 4. Методи виправлення граматичних помилок.**
- 5. Тензорні моделі природної мови.**
- 6. Моделі та методи сентимент аналізу.**
- 7. Методи генерації текстів. Нейронні мережі та структурні методи.**
- 8. Розв'язання неоднозначності слів. WSD.**
- 9. Автоматичне реферування текстів.**
- 10. Латентне розміщення Діріхле.**
- 11. Моделі векторизації семантики слів.**
- 12. Розпізнавання іменованих сущностей.**
- 13. Embeddings моделі представлення слів та словосполучень.**
- 14. Моделі представлення семантики речень.**
- 15. Задача визначення авторського стилю. Моделі та методи.**
- 16. Модель Glove.**
- 17. Information Retrieval.**
- 18. Пошукові індекси.**
- 19. Ранжоване отримання інформації.**
- 20. Архітектура пошукової системи.**
- 21. Методи прискорення пошуку текстів за запитом.**
- 22. Машинний переклад.**
- 23. Системи підтримки діалогу.**

- 24. Системи типу «Питання-Відповідь».**
- 25. Розпізнавання іменованих сущностей текстів за допомогою Умовних випадкових полів.**
- 26. Методи пониження розмірності латентних семантичних моделей.**
- 27. Методи автоматичної побудови таксономії.**
- 28. Формальний концептуальний аналіз.**
- 29. Лінгвістичні моделі.**
- 30. Генеративні лінгвістичні автомати.**

1. Архітектура системи лінгвістичного аналізу текстів

Архітектура системи – принципова організація системи, втілена в її елементах, їх взаємини один з одним і з середовищем, а також принципи, що направляють її проектування і еволюцію.

Поняття архітектури значною мірою суб'єктивне і має безліч суперечливих тлумачень; в кращому випадку воно відображає загальну точку зору команди розробників на результати проектування системи.

Система складається з блоку морфологічного лексичного аналізу, блоку синтаксичного аналізу та блоку семантичного аналізу.

Структурно система представляє собою послідовність блоків аналізу текстів, кожен з яких послідовно здійснює аналіз тексту. Речення тексту обробляються послідовно. На першому етапі роботи система за допомогою блоку морфологічного аналізу виконує лексико-морфологічний аналіз речення та визначає для кожного слова його нормальну форму та морфологічні характеристики (частина мови, рід, число, відмінок, час і т.д.). Наступним етапом аналізу є синтаксичний аналіз, який виконується блоком синтаксичного аналізу, що за даними попереднього етапу обробки вибудовує дерево підпорядкування (dependency tree) речення. Далі дані передаються на блок кореферентного аналізу, що розв'язує наступну задачу:

Блок морфологічного лексичного аналізу.

Вхідні дані: текст природною мовою

Обробка: блок розкладає текст на речення, речення на слова, потім обробляє кожне слово окремо. Функції морфологічного аналізу мають привести кожне слово до нормальної форми, та знайти його морфологічні характеристики

Вихід: текст, що складається із нормалізованих слів із визначеними морфологічними характеристиками.

Блок синтаксичного аналізу

Вхідні дані: розібраний морфологічно-нормалізований текст природною мовою.

Обробка: блок розбирає речення тексту, генеруючи дерева синтаксичного виводу речень.

Вихід: синтаксичні дерева речень тексту.

Блок семантичного аналізу

Вхідні дані: синтаксичні дерева речень тексту.

Обробка: функції семантичного аналізу розбирають синтаксичні дерева тексту за допомогою онтологічної бази знань та генерують семантичну структуру речень. Потім семантичні структури речень інтегруються в семантичний граф тексту. Блок семантичного аналізу вирішує проблеми полісемії, метонімії, заміни займенників та інші складні лінгвістичні проблеми неоднозначності.

Вихід: семантичний граф тексту, який обробляється функціями постсемантичного аналізу, що виконують такі прикладні задачі, як визначення тематики тексту, генерація реферату, смисловий переклад з однієї мови на іншу, підтримка природномовного діалогу з користувачем, забезпечення природномовного інтерфейсу для БД та багато інших.

Логічна архітектура підтримує функціонування системи протягом усього її життєвого циклу на логічному рівні. Вона складається з набору пов'язаних технічних концепцій і принципів. Логічна архітектура представляється за допомогою методів, що відповідають тематичним групам описів, і як мінімум, включає в себе функціональну архітектуру, поведінкову архітектуру і тимчасову архітектуру.

Мета проектування **фізичної архітектури** полягає в створенні фізичного, конкретного рішення, яке погоджено з логічною архітектурою і відповідає встановленим вимогам.

Після того, як логічна архітектура визначена, повинні бути ідентифіковані конкретні фізичні елементи, які підтримують функціональні, поведінкові, і тимчасові властивості, а також очікувані властивості системи, отримані з не функціональних вимог до системи.

2. Редакційна відстань.

https://uk.wikipedia.org/wiki/Відстань_Левенштейна

https://ru.wikipedia.org/wiki/Расстояние_Левенштейна

Відстань Левенштейна (також редакційна відстань) між двома рядками в теорії інформації та комп'ютерної лінгвістики – це мінімальна кількість операцій вставки одного символу, видалення одного символу і заміни одного символу на інший, необхідних для перетворення одного рядка в інший.

Відстань Левенштейна (також функція Левенштейна, алгоритм Левенштейна або відстань редагування) у [теорії інформації і комп'ютерній лінгвістиці](#) міра відмінності двох послідовностей символів (рядків). Обчислюється як мінімальна кількість операцій вставки, видалення і заміни, необхідних для перетворення одної послідовності в іншу.

Приклад:

Щоб перетворити слово **небо** на слово **треба** необхідно зробити дві заміни та одну вставку, відповідно дистанція Левенштейна становить 3:

1. **небо** -> **неба** (замінюємо о на а)
2. **неба** -> **реба** (замінюємо н на р)
3. **реба** -> **треба** (вставляємо т)

На практиці дистанція Левенштейна використовується для визначення подібності послідовностей символів, наприклад для корекції орфографії або для пошуку дублікатів.

Применение

Расстояние Левенштейна и его обобщения активно применяется:

- для исправления ошибок в слове (в поисковых системах, базах данных, при вводе текста, при автоматическом распознавании отсканированного текста или речи).

- для сравнения текстовых файлов утилитой `diff` и ей подобными. Здесь роль «символов» играют строки, а роль «строк» — файлы.
- в биоинформатике для сравнения [генов, хромосом и белков](#).

С точки зрения приложений определение расстояния между словами или текстовыми полями по Левенштейну обладает следующими недостатками:

1. При перестановке местами слов или частей слов получаются сравнительно большие расстояния;
2. Расстояния между совершенно разными короткими словами оказываются небольшими, в то время как расстояния между очень похожими длинными словами оказываются значительными.

Редакционное предписание

Редакционным предписанием называется последовательность действий, необходимых для получения из первой строки второй кратчайшим образом. Обычно действия обозначаются так: **D** ([англ. delete](#)) — удалить, **I** ([англ. insert](#)) — вставить, **R** ([replace](#)) — заменить, **M** ([match](#)) — совпадение.

Разные цены операций

Цены операций могут зависеть от вида операции (вставка, удаление, замена) и/или от участвующих в ней символов, отражая разную вероятность мутаций в биологии^[3], разную вероятность разных ошибок при вводе текста и т. д.

В общем случае:

- $w(a, b)$ — цена замены символа a на символ b
- $w(\epsilon, b)$ — цена вставки символа b
- $w(a, \epsilon)$ — цена удаления символа a

Необходимо найти последовательность замен, минимизирующую суммарную цену.

Расстояние Левенштейна является частным случаем этой задачи при

- $w(a, a) = 0$
- $w(a, b) = 1$ при $a \neq b$

- $w(\varepsilon, b) = 1$
- $w(a, \varepsilon) = 1$

Здесь и ниже мы считаем, что все w неотрицательны, и действует правило треугольника: если две последовательные операции можно заменить одной, это не ухудшает общую цену (например, заменить символ x на y , а потом с y на z не лучше, чем сразу x на z).

Формула [править | править код]

Здесь и далее считается, что элементы строк нумеруются с первого, как принято в математике, а не с нулевого, как принято в некоторых языках программирования.

Пусть S_1 и S_2 — две строки (длиной M и N соответственно) над некоторым [алфавитом](#), тогда редакционное расстояние (расстояние Левенштейна) $d(S_1, S_2)$ можно подсчитать по следующей [рекуррентной формуле](#)

$$d(S_1, S_2) = D(M, N), \text{ где}$$

$$D(i, j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ \min\{ & , \\ D(i, j - 1) + 1, & j > 0, i > 0 \\ D(i - 1, j) + 1, & \\ D(i - 1, j - 1) + m(S_1[i], S_2[j]) & \} \end{cases}$$

где $m(a, b)$ равна нулю, если $a = b$ и единице в противном случае;
 $\min\{a, b, c\}$ возвращает наименьший из аргументов.

Здесь шаг по i символизирует удаление (D) из первой строки, по j — вставку (I) в первую строку, а шаг по обоим индексам символизирует замену символа (R) или отсутствие изменений (M).

Очевидно, справедливы следующие утверждения:

- $d(S_1, S_2) \geq |S_1| - |S_2|$
- $d(S_1, S_2) \leq \max(|S_1|, |S_2|)$
- $d(S_1, S_2) = 0 \Leftrightarrow S_1 = S_2$

3. Методи виправлення орфографічних помилок.

<http://www.lrec-conf.org/proceedings/lrec2000/pdf/221.pdf>

Метод частоти слів

З метою вибору кращої корекції, LOB-корпус (Atwell i Elliott, 1987) використовується для частотної інформації слова. Наприклад, помилкове слово "whith" породило можливості ("with" 7201), ("which" 4467), ("white" 261), and ("whit" 0) — де цифрами є частота пов'язані слова в корпусі. Найбільш частим словом серед можливих поправок для "whith" (а отже і кращої корекції за цим методом) є, таким чином, "with". Зверніть увагу, що "whit" - це правильне слово, яке не зустрічається в корпусі LOB. Якщо синтаксична інформація застосовується до наведеного вище прикладу (наприклад, "whith" ADJ), то буде запропоновано лише одну можливість (наприклад, "білий" 261). Таким чином, використання синтаксичної інфекції підвищує ефективність коректування орфографії.

Метод відстані символів

Цей метод використовує метрику типу Піфагора для вимірювання відстані між словом з помилковим написанням і можливим виправленням, заснованим на розкладці клавіатури Qwerty (Min i Wilson, 1995). Клавіатура Qwerty представлена двома двовимірними масивами: одна для клавіш нижнього регістру, а інша для клавіш верхнього регістру (рис. 1).

i\j	0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	0	-	=
1	q	w	e	r	t	Y	u	i	o	p	[]
2	a	s	d	f	g	H	j	k	l	;	'	
3	z	x	c	v	b	N	m	,	.	/		

A. Редакційна відстань

метод базується на припущення, що людина зазвичай робить кілька помилок, якщо мінімальне число основних операцій редагування (вставлення, видалення,

підстановки), необхідні для приховування слова словника до неслова. Чим нижче, тим більше, тим вище ймовірність

Користувач зробив такі помилки. Змінити відстань корисно для виправлення помилок, що виникають в результаті введення з клавіатури, оскільки вони є

часто такого ж типу, як і дозволені операції редагування. Це не зовсім так добре для коригування фонетичних помилок орфографії, особливо якщо різниця між написанням і вимовою велика, як англійською або французькою.

В. Ключі подібності

Ключу, призначенному кожному словнику словника, порівнюють лише ключі словника з ключем, який обчислюється для цього слова Слово, для якого клавіші обчислюються для не слова. Слово, для якого найбільш близькі клавіші, вибираються як Такий підхід є швидкісним, оскільки з гарним потрібно обробляти тільки слова з подібними ключами Алгоритм перетворення цього методу може обробляти помилки клавіатури

С. Методи, засновані на правилах

Методи, засновані на правилах, цікаві. Вони працюють за допомогою набору правил, які охоплюють поширені орфографічні та друкарські помилки

і застосування цих правил до слова з помилкою. Інтуїтивно ці правила є «оберненими» поширеними помилками. Кожен правильний слово, що генерується цим процесом, приймається як пропозиція корекції. Правила також мають ймовірності, що дозволяють ранжувати пропозиції шляхом накопичення ймовірностей для застосованих правил. Змінити відстань можна розглядати як особливий випадок а

метод, заснований на правилах, з обмеженням на можливі правила.

D. Методи на основі N-грамів

N-грами можна використовувати двома способами, або без словника, або разом з словником. Використовується без словника, ngrams використовуються, щоб знайти, в якій позиції в помилковому слові виникає помилка. Якщо є унікальний спосіб змінити неправильно написане слово так, що воно містить тільки допустимі n-грами, це приймається як корекція. Виконання цього методу обмежені. Його головна чеснота полягає в тому, що вона проста і не вимагає якого-небудь словника. Разом з словником звички використовувати n-грами

Визначте відстань між словами, але слова завжди перевіряються проти словника. Це можна зробити кількома способами, наприклад, перевірте, скільки n-грам слів із помилковим словом і словником словника мають спільне значення, зважені довжиною слова.

E. Ймовірнісні методи

Вони, просто кажучи, засновані на деяких статистичних ознаках мови. Двома поширеними методами є ймовірності переходу і ймовірності плутанини. Ймовірності переходу подібні до n-грамів. Вони дають нам ймовірність того, що даний лист або послідовність літер супроводжується іншим даним літером. Ймовірності переходу не дуже корисні, коли ми маємо доступ словник або індекс. Враховуючи пропозицію, яку слід виправити, система розкладає кожен рядок у реченні на букви n-грами і отримує слова-кандидати з лексикону шляхом порівняння рядків n-грамів з n-грамами вводу лексикону. Отримано кандидати ранжируються за умовою ймовірністю збігів з рядком, з урахуванням ймовірностей заміщення символів. Нарешті, для визначення найкращої послідовності висловлювань слова у реченні використовують модель слова-біграм і певний алгоритм. Вони стверджують, що система може виправляти помилки, які не стосуються слова, а також реальні помилки в словах і досягає 60,2% зниження помилки для реального тексту розпізнавання.

F. Нейронні мережі

Нейронні мережі також є цікавою і перспективною технікою, але здається, що вона повинна дозріти трохи більше, перш ніж вона може бути

використовуються в цілому. Поточні методи базуються на мережах зворотного поширення, використовуючи один вихідний вузол для кожного слова в

словник і вхідний вузол для кожного можливого n-граму в кожному положенні слова, де n зазвичай одно або два.

Зазвичай тільки один з виходів повинен бути активним, що вказує, які слова словника пропонує мережа як корекція. Цей метод працює для малих (<1000 слів) словників, але він не масштабується добре. Вимоги часу є великими традиційного обладнання, особливо в фазі навчання.

4. Методи виправлення граматичних помилок.

Граматична корекція помилок (GEC) є завданням виправлення різних видів помилок у тексті, таких як орфографічні, пунктуаційні, граматичні та помилки вибору слова.

GEC зазвичай формулюється як завдання виправлення речення. Система GEC приймає потенційно помилкове речення в якості вхідних даних і, як очікується, перетворить його на свою виправлену версію. Див. Приклад, наведений нижче:

Input (Erroneous)	Output (Corrected)
She see Tom is caught by policeman in park at last night.	She saw Tom caught by a policeman in the park last night.

seq2seq для GEC

Більшість моделей seq2seq для GEC мають два недоліки.

- По-перше, моделі seq2seq навчаються лише з обмеженими поправками з поправками, як показано на рис. 1 (a). Обмежені розмірами навчальних даних, моделі з мільйонами параметрів можуть бути не узагальнені. Таким чином, звичайно, що моделі не в змозі виправити речення ідеально, навіть якщо пропозиція трохи відрізняється від екземпляра навчання, як показано на малюнку 1 (b).
- По-друге, моделі seq2seq зазвичай не можуть повністю виправити речення з багатьма граматичними помилками через однократне виведення seq2seq, як показано на малюнку 1 (b) і 1 (c), тому що деякі помилки у реченні можуть зробити контекст дивним, що заплутує моделей для виправлення інших помилок.

В качестве нейронного машинного перевода (NMT), типичный нейронный подход GEC использует модель seq2seq кодер-декодер с механизмом внимания для редактирования необработанного предложение в грамматически правильное предложение

Given a raw sentence $\mathbf{x}^r = (x_1^r, \dots, x_M^r)$ and its corrected sentence $\mathbf{x}^c = (x_1^c, \dots, x_N^c)$ in which x_M^r and x_N^c are the M -th and N -th words of sentence \mathbf{x}^r and \mathbf{x}^c respectively, the error correction seq2seq model learns a probabilistic mapping $P(\mathbf{x}^c|\mathbf{x}^r)$ from error-corrected sentence pairs through maximum likelihood estimation (MLE), which learns model parameters Θ_{crt} to maximize the following equation:

$$\Theta_{crt}^* = \arg \max_{\Theta_{crt}} \sum_{(\mathbf{x}^r, \mathbf{x}^c) \in \mathcal{S}^*} \log P(\mathbf{x}^c|\mathbf{x}^r; \Theta_{crt}) \quad (1)$$

where \mathcal{S}^* denotes the set of error-corrected sentence pairs.

For model inference, an output sequence $\mathbf{x}^o = (x_1^o, \dots, x_i^o, \dots, x_L^o)$ is selected through beam search, which maximizes the following equation:

$$P(\mathbf{x}^o|\mathbf{x}^r) = \prod_{i=1}^L P(x_i^o|\mathbf{x}^r, \mathbf{x}^o_{<i}; \Theta_{crt}) \quad (2)$$

Для нейронного GEC его целью является улучшение беглости предложения4 без изменения его первоначального значения; таким образом, любая пара предложений, которая удовлетворяет этому условию (мы называем это условием повышения беглости), может использоваться в качестве обучающего экземпляра.

In this work, we define $f(\mathbf{x})$ as the fluency score of a sentence \mathbf{x} :

$$f(\mathbf{x}) = \frac{1}{1 + H(\mathbf{x})} \quad (3)$$

$$H(\mathbf{x}) = -\frac{\sum_{i=1}^{|\mathbf{x}|} \log P(x_i|\mathbf{x}_{<i})}{|\mathbf{x}|} \quad (4)$$

where $P(x_i|\mathbf{x}_{<i})$ is the probability of x_i given context $\mathbf{x}_{<i}$, computed by a language model, and $|\mathbf{x}|$ is the length of sentence \mathbf{x} . $H(\mathbf{x})$ is actually the cross entropy of the sentence \mathbf{x} , whose range is $[0, +\infty)$. Accordingly, the range of $f(\mathbf{x})$ is $(0, 1]$.

В обучении, улучшающем беглость речи, не только модель seq2seq обучается с исходными парами предложений с исправленными ошибками, но также генерирует менее беглые предложения (например, из своих n -лучших выходных данных) для создания новых пар предложений с исправленными ошибками путем их сопряжения с их правильные предложения во время обучения, пока беглость предложений1 ниже, чем у их правильных предложений, как показано на рисунке 2 (а). В частности, мы называем сгенерированные пары предложений с исправленными ошибками беглыми предложениями,

потому что предложение на целевой стороне всегда улучшает беглость по сравнению с исходной. Сгенерированные пары предложений повышения беглости речи во время обучения будут использоваться в качестве дополнительных обучающих примеров в течение последующих эпох обучения, позволяя модели исправления ошибок видеть более грамматически неправильные предложения во время обучения и, соответственно, улучшая ее способность к обобщению.

Algorithm 1 Back-boost learning

- 1: Train error generation model Θ_{gen} with $\widetilde{\mathcal{S}^*}$;
- 2: **for** each sentence pair $(\mathbf{x}^r, \mathbf{x}^c) \in \mathcal{S}$ **do**
- 3: Compute $\mathcal{D}_{back}(\mathbf{x}^c)$ according to Eq (5);
- 4: **end for**
- 5: **for** each training epoch t **do**
- 6: $\mathcal{S}' \leftarrow \emptyset$;
- 7: Derive a subset \mathcal{S}_t by randomly sampling $|\mathcal{S}^*|$ elements from \mathcal{S} ;
- 8: **for** each $(\mathbf{x}^r, \mathbf{x}^c) \in \mathcal{S}_t$ **do**
- 9: Establish a fluency boost pair $(\mathbf{x}', \mathbf{x}^c)$ by randomly sampling $\mathbf{x}' \in \mathcal{D}_{back}(\mathbf{x}^c)$;
- 10: $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(\mathbf{x}', \mathbf{x}^c)\}$;
- 11: **end for**
- 12: Update error correction model Θ_{crt} with $\mathcal{S}^* \cup \mathcal{S}'$;
- 13: **end for**

Основываясь на идее многоокруглой коррекции, мы также предлагаем продвинутый подход к увеличению беглости речи: круговое исправление ошибок. Вместо постепенного исправления предложения с помощью той же модели seq2seq, которая была представлена в Разделе 4.1, круговое исправление последовательно корректирует предложение с помощью модели seq2seq справа налево и модели seq2seq слева направо, как показано на рисунке 4.

Мотивация обходного исправления ошибок проста. Декодеры с разными порядками декодирования декодируют последовательности слов в разных контекстах, благодаря чему они имеют свои уникальные преимущества для конкретных типов ошибок. Для примера на рисунке 4 ошибка отсутствия статьи (например, парк → парк), скорее всего, будет исправлена моделью seq2seq справа налево, чем моделью слева направо, потому что добавление статьи зависит от существительного парка, который уже был замечен моделью справа налево, когда он принимал решение. Напротив, модель слева направо могла бы лучше справляться с ошибками согласования глагола субъекта (например, come → приходит на рисунке 4), потому что ключевое слово, которое

решает форму глагола, является его субъектом She, который находится в начале предложения.

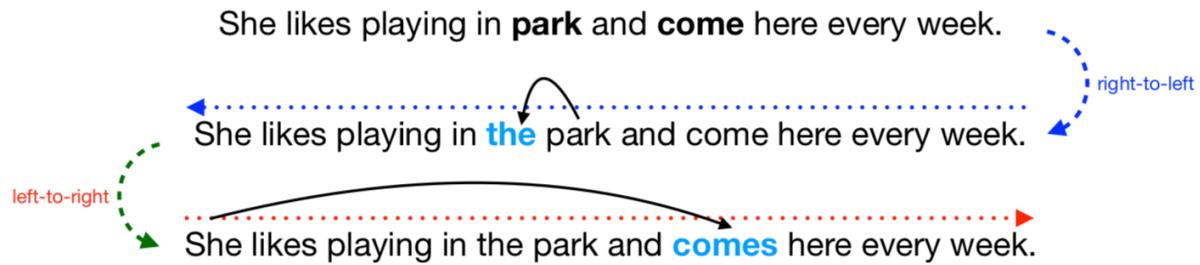


Figure 4: Round-way error correction: some types of errors (e.g., articles) are easier to be corrected by a right-to-left seq2seq model, while some (e.g., subject verb agreement) are more likely to be corrected by a left-to-right seq2seq model. Round-way error correction makes left-to-right and right-to-left seq2seq models well complement each other, enabling it to correct more grammatical errors than an individual model.

5. Тензорні моделі природної мови.

<https://pdfs.semanticscholar.org/8393/>

[dcbfc2ca1216e4810a2aea62eaca16ba1d3b.pdf](https://pdfs.semanticscholar.org/8393/dcbfc2ca1216e4810a2aea62eaca16ba1d3b.pdf)

<https://www.aclweb.org/anthology/N18-1114>

Tensor Product Generation — это общий метод генерации вложений в векторное пространство сложных символьных структур.

Центральная идея TPR (Смоленский, 1990) может быть оценена путем сопоставления TPR для строки слова с вложением в векторное пространство пакета слов (BoW).

The central idea of TPRs (Smolensky, 1990) can be appreciated by contrasting the TPR for a word string with a bag-of-words (BoW) vector-space embedding.

In a BoW embedding, the vector that encodes Jay saw Kay is the same as the one that encodes Kay saw Jay: $J + K + s$ where J, K, s are respectively the vector embeddings of the words Jay, Kay, saw.

A TPR embedding that avoids this confusion starts by analyzing Jay saw Kay as the set $\{J\text{ay/SUBJ}, K\text{ay/OBJ}, s\text{aw/VERB}\}$. (Other analyses are possible: see Section 3.) Next we choose an embedding in a vector space V_F for Jay, Kay, saw as in the BoW case: J, K, s . Then comes the step unique to TPRs: we choose an embedding in a vector space V_R for the *roles* SUBJ, OBJ, VERB: $r_{\text{SUBJ}}, r_{\text{OBJ}}, r_{\text{VERB}}$. Crucially, $r_{\text{SUBJ}} \neq r_{\text{OBJ}}$. Finally, the TPR for Jay saw Kay is the following vector in $V_F \otimes V_R$:

$$\mathbf{v}_{\text{Jay saw Kay}} = J \otimes r_{\text{SUBJ}} + K \otimes r_{\text{OBJ}} + s \otimes r_{\text{VERB}} \quad (1)$$

Each word is tagged with the role it fills in the sentence; Jay and Kay fill different roles.

Связанные со структуралистскими идеями лингвистики, исследователи утверждают, что значение слова можно смоделировать, сравнивая распределение слов в тексте (Schütze, 1993). Популярный подход к представлению этих распределений слов заключается в сборе частот встречаемости слов и их размещении в многомерных контекстных векторах (Turney and Pantel, 2010). Этот подход позволяет использовать методы линейной алгебры для моделирования отношений между объектами, включая семантические ассоциации, в геометрическом пространстве.

Двумя наиболее известными семантическими пространственными моделями в литературе являются HAL (гиперпространственный аналог языка; Lund and Burgess (1996)) и LSA (латентный семантический анализ; Landauer and Dumais (1997)). Эти две модели отличаются тем, как они строят свои контекстные векторы. HAL строит контекстные векторы, сохраняя частоты встречаемости слов до и после заказа в пословной матрице. Рассмотрим матрицу HAL, показанную в таблице 1, созданную предложением «собачий бит почтальона» с использованием скользящего контекстного окна с радиусом 2. Информация о совместном вхождении, предшествующем и последующем заполнении каждого слова, записывается отдельно строкой и векторы столбцов.

LSA отличается от HAL тем, что векторы контекста LSA формируются путем сбора частот встречаемости слова в каждом документе для создания матрицы слова-документа. Затем используется дорогостоящий метод, известный как разложение по одному значению (SVD), чтобы уменьшить размеры матрицы текстового документа до k наиболее значимых скрытых понятий.

Модель тензорного кодирования (TE) строит свое семантическое пространство, используя эффективный процесс связывания, основанный на произведениях Кронекера теоретически

неограниченных единичных векторов. Результат содержит информацию о контексте и порядке в одном представлении, которое мы называем тензор памяти.

Модель невід'ємної факторизації тензорів, у якій зберігається дані про частоту використання різних словосполучень у великих текстових корпусах із врахуванням синтаксичної позиції слів. Факторизовані тензорні лінгвістичні моделі дозволяють автоматизовано виділяти з корпусів текстів опис таких лінгвістичних структур, як селективні преференції (selectional preferences) та субкатегоріальні фрейми дієслів [2], що містять дані про семантичні та синтаксичні властивості зв'язків між дієсловами та їх аргументами-іменниками у реченнях.

Тривимірний тензор для зберігання частотних оцінок сполучностей слів – підметів, присудків та прямих додатків, отриманих в результаті аналізу великих текстових корпусів, зображений на рисунку 1.

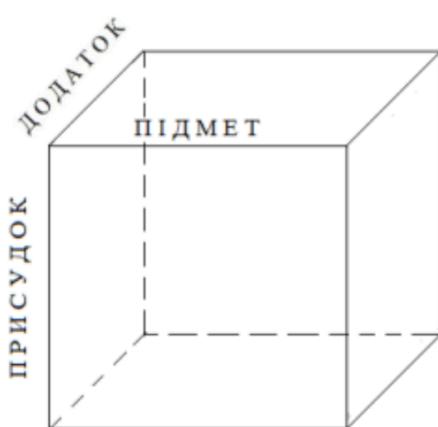


Рисунок 1. Тривимірний тензор для зберігання частотних даних сполучностей слів – підметів, присудків та прямих додатків.

В результаті частотного аналізу текстових корпусів формується розріджений тензор великої розмірності. З метою отримання більш стислого та зручного представлення до тензору застосовується невід'ємна тензорна факторизація.

Основна ідея методу полягає в мінімізації суми квадратів різниць значень між оригіналом тензору та його факторизованою моделлю. Для тривимірного випадку тензору $T \in R^{D1 \times D2 \times D3}$ це відповідає рівнянню

$$\min_{x_i \in R^{D1}, y_i \in R^{D2}, z_i \in R^{D3}} \| T - \sum_{i=1}^k x_i \circ y_i \circ z_i \|_F^2 , \text{ де } k -$$

число вимірів у факторизованій моделі.

При невід'ємній тензорній факторизації додається обмеження невід'ємності, перетворюючи модель у

$$\min_{x_i \in R_{\geq 0}^{D1}, y_i \in R_{\geq 0}^{D2}, z_i \in R_{\geq 0}^{D3}} \| T - \sum_{i=1}^k x_i \circ y_i \circ z_i \|_F^2$$

В результаті факторизації кожен підмет-іменник, присудок-дієслово та додаток-іменник отримує власний вектор розмірності k з відповідних матриць.

Оригінальне значення з тензору T для трійки (s, v, o) x_{svo} може бути відновлене у факторизованій моделі обчисленням суми

$$x_{svo} = \sum_{i=1}^k s_{si} v_{vi} o_{oi} .$$

Вектори з матриць факторизованого тензору описують комутативні властивості слів.

6. Моделі та методи сентимент аналізу.

- ◎ <https://monkeylearn.com/sentiment-analysis/>
- ◎ https://ru.wikipedia.org/wiki/Анализ_тональности_текста
- ◎ <https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17>
- ◎ <https://liris.cnrs.fr/Documents/Liris-6508.pdf>

Аналіз тональності тексту (сентимент-аналіз) – клас методів контент-аналізу в комп'ютерній лінгвістиці, призначений для автоматизованого виявлення в текстах емоційно забарвленої лексики і емоційної оцінки авторів (думок) по відношенню до об'єктів, мова про які йде в тексті.

Тональність – це емоційне ставлення автора висловлювання до деякого об'єкту (об'єкту реального світу, події, процесу або їх властивостями / атрибутами), виражене в тексті. Емоційна складова, виражена на рівні лексеми або комунікативного фрагмента, називається лексичною тональністю (або лексичним сентиментом). Тональність всього тексту в цілому можна визначити як функцію (в найпростішому випадку суму) лексичних тональностей складових його одиниць (пропозицій) і правил їх поєднання.

Анализ настроений, также известный как Opinion Mining, является областью в Natural Language Processing (NLP), которая создает системы, которые пытаются идентифицировать и извлекать мнения в тексте. Обычно, помимо определения мнения, эти системы извлекают атрибуты выражения, например:

- ◎ **Полярность:** если выступающий выражает положительное или отрицательное мнение,
- ◎ **Предмет:** о чем идет речь,
- ◎ **Держатель мнения:** физическое или юридическое лицо, выражающее мнение.

Текстовая информация может быть разделена на два основных типа: факты и мнения.

Факты — это объективные выражения чего-то.

Мнения, как правило, являются субъективными выражениями, которые описывают чувства, оценки и чувства людей по отношению к предмету или теме.

Анализ настроений, как и многие другие проблемы НЛП, может быть смоделирован как проблема классификации, где необходимо решить две подзадачи:

- Классификация предложения как субъективного или объективного, известного как субъективная классификация.
- Классификация предложения как выражения положительного, отрицательного или нейтрального мнения, известного как классификация полярности.

Анализ настроений может применяться на разных уровнях охвата:

- Анализ настроений на уровне документа позволяет получить представление о полном документе или абзаце.
- Анализ настроения на уровне предложения позволяет получить представление об одном предложении.
- Анализ настроения на уровне под-предложения позволяет получить представление о под-выражениях в предложении.

Методы классификации тональности

Методы, основанные на правилах и словарях

Этот метод основан на поиске эмотивной лексики^[39] (лексической тональности) в тексте по заранее составленным тональным словарям и

правилам с применением лингвистического анализа. По совокупности найденной эмотивной лексики текст может быть оценен по шкале, содержащей количество негативной и позитивной лексики. Данный метод может использовать как списки правил, подставляемые в регулярные выражения, так и специальные правила соединения тональной лексики внутри предложения. Чтобы проанализировать текст, можно воспользоваться следующим алгоритмом: сначала каждому слову в тексте присвоить его значение тональности из словаря (если оно присутствует в словаре), а затем вычислить общую тональность всего текста путём суммирования значения тональностей каждого отдельного предложения^[39].

Метод, основанный на теоретико-графовых моделях

В основе этого метода используется предположение о том, что не все слова в [текстовом корпусе](#) документа равнозначны. Какие-то слова имеют больший вес и сильнее влияют на тональность текста. При использовании этого метода анализ тональности разбивается на несколько этапов:

1. построение графа на основе исследуемого текста;
2. ранжирование его вершин;
3. классификация найденных слов;
4. вычисление результата.

Машинное обучение с учителем

В наше время наиболее часто используемыми в исследованиях методами являются методы на основе [машинного обучения](#) с учителем. Сутью таких методов является то, что на первом этапе обучается машинный классификатор (например, [байесовский](#)^[41]) на заранее размеченных текстах, а затем используют полученную модель при анализе новых документов. Опишем краткий алгоритм^[42]:

1. вначале собирается коллекция документов, на основе которой обучается машинный классификатор;
2. каждый документ раскладывается в виде вектора признаков(аспектов), по которым он будет исследоваться;
3. указывается правильный тип тональности для каждого документа;
4. производится выбор алгоритма классификации и метод для обучения классификатора;

5. полученную модель используем для определения тональности документов новой коллекции.

Типы анализа настроений (Types of Sentiment Analysis)

Существует много типов и разновидностей анализа настроений, и инструменты SA варьируются от систем, которые фокусируются на полярности (положительная, отрицательная, нейтральная), до систем, которые обнаруживают чувства и эмоции (сердитый, счастливый, грустный и т. д.) Или определяют намерения (например, заинтересованные v. не интересно). В следующем разделе мы рассмотрим наиболее важные из них.

Детальный анализ настроений

Иногда вам также может быть интересно более точно определить уровень полярности мнений, поэтому вместо того, чтобы просто говорить о положительных, нейтральных или отрицательных мнениях, вы можете рассмотреть следующие категории:

- Очень позитивный
- положительный
- нейтральный
- отрицательный
- Очень негативно

Обычно это называется детальным анализом настроений. Это может быть, например, отображено на 5-звездочный рейтинг в обзоре, например: Очень положительный = 5 звезд и Очень отрицательный = 1 звезда.

Некоторые системы также обеспечивают различные разновидности полярности, определяя, связаны ли положительные или отрицательные чувства с определенным чувством, таким как гнев,

грусть или беспокойство (то есть отрицательные чувства) или счастье, любовь или энтузиазм (то есть положительные чувства).

Обнаружение эмоций

Обнаружение эмоций направлено на выявление таких эмоций, как счастье, разочарование, гнев, грусть и тому подобное. Многие системы обнаружения эмоций используют лексиконы (то есть списки слов и эмоций, которые они передают) или сложные алгоритмы машинного обучения.

Одним из недостатков использования лексиконов является то, что способ выражения людьми своих эмоций сильно отличается, также как и лексические элементы, которые они используют. Некоторые слова, которые обычно выражают гнев, такие как дермо или убивают (например, в вашем продукте есть кусок дерма или ваша служба поддержки убивает меня), также могут выражать счастье (например, в таких текстах, как Это дермо или Вы убиваете его).

Аспектный анализ настроений

Обычно при анализе настроений по предметам, например, по продуктам, вас может интересовать не только то, говорят ли люди с положительной, нейтральной или отрицательной полярностью о продукте, но также и о конкретных аспектах или особенностях продукта, о которых говорят люди. , Вот что такое аспектный анализ настроений. В нашем предыдущем примере:

«Время автономной работы этой камеры слишком короткое».

Предложение выражает негативное мнение о камере, а точнее, о времени автономной работы, что является характерной особенностью камеры.

Анализ намерений

Анализ намерений в основном определяет, что люди хотят делать с текстом, а не то, что люди говорят с этим текстом. Посмотрите на следующие примеры:

«Ваша служба поддержки - это катастрофа. Я был в ожидании в течение 20 минут».

«Я хотел бы знать, как заменить картридж».

«Можете ли вы помочь мне заполнить эту форму?»

У человека нет проблем с обнаружением жалобы в первом тексте, вопроса во втором тексте и запроса в третьем тексте. Тем не менее, машины могут иметь некоторые проблемы для их идентификации. Иногда предполагаемое действие может быть выведено из текста, но иногда для его вывода требуются некоторые контекстуальные знания.

Многоязычный анализ настроений

Многоязычный анализ настроений может быть сложной задачей. Обычно требуется много предварительной обработки, и эта предварительная обработка использует ряд ресурсов. Большинство из этих ресурсов доступны в Интернете (например, лексиконы настроений), но необходимо создать много других (например,

переведенные корпуса или алгоритмы обнаружения шума). Использование доступных ресурсов требует большого опыта программирования и может занять много времени для реализации.

Альтернативой этому было бы автоматическое определение языка в текстах, затем подготовка пользовательской модели для языка по вашему выбору (если тексты не написаны на английском языке) и, наконец, выполнение анализа.

7. Методи генерації текстів. Нейронні мережі та структурні методи

https://ru.wikipedia.org/wiki/Генератор_текста

<https://arxiv.org/pdf/1711.09534.pdf>

- Structural?: <http://wing.comp.nus.edu.sg/~antho/P/P84/P84-1076.pdf>
- Structural?: <https://habr.com/ru/post/163727/>
- Markov: http://www.manhunter.ru/webmaster/358_generator_teksta_na_osnove_serey_markova.html
- LSTM: <https://www.kaggle.com/shivamb/beginners-guide-to-text-generation-using-lstms>
- RNN: https://www.tensorflow.org/tutorials/sequences/text_generation
- RNN: <https://towardsdatascience.com/generating-text-using-a-recurrent-neural-network-1c3bfee27a5e>
- GAN: <https://www.topbots.com/ai-research-gan-vae-text-generation/>
- GAN: <https://becominghuman.ai/generative-adversarial-networks-for-text-generation-part-1-2b886c8cab10>

Стандартный автоэнкодер состоит из двух нейронных сетей:

- кодер, состоящий из сверточных слоев, которые кодируют объект (изображение, текст, звук) в скрытый вектор; а также
- декодер, состоящий из деконволюционных слоев, которые декодируют скрытый вектор обратно в объект.

LSTM для генерации текста

В отличие от нейронных сетей с прямой связью, в которых выходы активации распространяются только в одном направлении, выходы активации нейронов распространяются в обоих направлениях (от входов к выходам и от выходов к входам) в рекуррентных нейронных сетях. Это создает петли в архитектуре нейронной сети, которая действует как «состояние памяти» нейронов. Это состояние позволяет нейронам запоминать то, что было изучено до сих пор.

Состояние памяти в RNN дает преимущество над традиционными нейронными сетями, но с ними связана проблема, называемая исчезающим градиентом. В этой задаче при обучении с большим количеством слоев сети становится действительно трудно изучать и настраивать параметры более ранних уровней. Для решения этой проблемы был разработан новый тип сетей RNN, называемых моделями LSTM (долговременная кратковременная память).

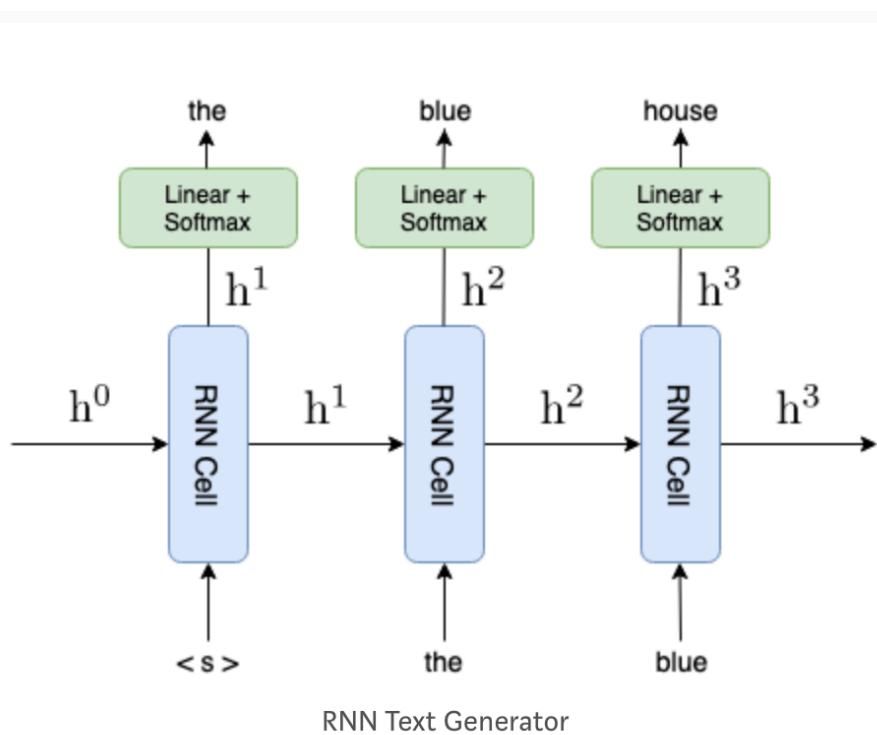
LSTM имеют дополнительное состояние, называемое «состоянием ячейки», через которое сеть вносит корректизы в информационный поток. Преимущество этого состояния состоит в том, что модель может помнить или забывать склонности более избирательно. Чтобы узнать больше о LSTM, вот отличный пост. Позволяет архитектуре модели LSTM в нашем коде. Я добавил всего три слоя в модель.

Входной слой: принимает последовательность слов в качестве ввода

LSTM Layer: вычисляет вывод с использованием единиц LSTM. Я добавил 100 единиц в слой, но это число можно настроить позже.

Dropout Layer: слой регуляризации, который случайным образом отключает активации некоторых нейронов в слое LSTM. Это помогает в предотвращении переоснащения. (Необязательный слой)

Выходной слой: вычисляет вероятность наилучшего возможного следующего слова в качестве выходного



GAN

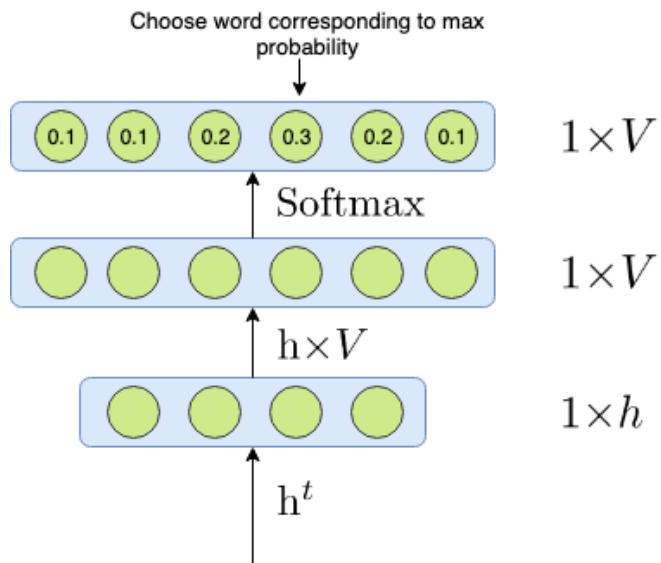
генерация текста с помощью простого генератора текста на основе RNN.

На каждом временном шаге t RNN принимает ранее сгенерированный токен и предыдущее скрытое состояние в качестве входных данных и генерирует новое скрытое состояние h^t .

$$h^t = RNN(h^{t-1}, w^{t-1})$$

Затем скрытое состояние пропускается через линейный слой и слой softmax, за которым следует argmax, чтобы получить следующее слово.

RNN обучается, заставляя его предсказывать следующее слово в предложении на каждом временном шаге. Обучение выполняется путем обратного распространения кросс-энтропийной потери между выходным распределением слоя softmax и целевым вектором с одним горячим током.



Теперь рассмотрим этот генератор на основе RNN как генераторную сеть в GAN. Здесь скрытый вектор z является скрытым входным состоянием h^0 RNN, а выходной сигнал $G(z)$ генератора является выходным предложением RNN. Разница здесь в том, что вместо того, чтобы обучать RNN минимизировать кросс-энтропийные потери по отношению к целевым горячим векторам, мы будем тренировать его, чтобы увеличить вероятность того, что сеть дискриминатора классифицирует предложение как «реальное», то есть цель теперь состоит в том, чтобы минимизировать $1 - D(G(z))$.

Помните, что при декодировании с использованием RNN на каждом временном шаге мы делаем выбор следующего слова, выбирая слово, соответствующее максимальной вероятности, из вывода функции softmax. Эта операция «подбора» недифференцируема.

Почему это проблема? Это проблема, потому что, чтобы обучить генератор минимизировать $1 - D(G(z))$, нам нужно подать выход генератора на дискриминатор и распространить обратно соответствующую потерю дискриминатора. Чтобы эти градиенты достигли генератора, они должны пройти через недифференцируемую операцию «выбора» на выходе генератора. Это проблематично, так как обратное распространение зависит от дифференцируемости всех уровней в сети.

8. Розв'язання неоднозначності слів. WSD.

<https://ru.wikipedia.org/wiki/>

Разрешение лексической многозначности

Разрешение лексической многозначности (word sense disambiguation, WSD) — это неразрешенная проблема [обработки естественного языка](#), которая заключается в задаче выбора значения (или смысла) [многозначного слова](#) или словосочетания в зависимости от [контекста](#), в котором оно находится. Данная задача возникает в [дискурсивном анализе](#), при оптимизации [релевантности](#) результатов поисковыми системами, при разрешении [анафорических ссылок](#), в исследовании лингвистической [когерентность](#) текста, при анализе [умозаключений](#).

О чём речь, можно понять из следующего примера с неоднозначным словом «ключ»:

1. ключ как инструмент для открывания
2. ключ как источник воды

а также 3 контекста:

1. Ключ подошёл, дверь открылась
2. Я напился из ключа
3. Жизнь бьёт ключом

Проблемы и трудности

Составление словарей

Во-первых, все словари разные и не эквивалентны друг другу. Чаще всего задача отличить смыслы слова друг от друга не вызывает трудностей, однако в некоторых случаях различные значения слова могут быть очень близкими друг другу семантически (например, если каждый из них является метафорой или метонимией друг к другу), и в таких ситуациях разделение на смыслы в разных [словарях](#) и [тезаурусах](#) может значительно различаться.

Определение части речи

Во-вторых, в некоторых языках **проблема определения части речи** (англ. *Part-of-speech tagging*) слова может быть очень близко связана с проблемой разрешения многозначностей, в результате чего эти две задачи могут друг другу мешать.

Человеческий фактор и согласованность ручных результатов

Здравый смысл

Некоторые исследователи утверждают^[10], что при обработке текстов немаловажен также **здравый смысл**, обучить которому компьютер представляется маловозможным. В качестве примера можно привести два следующих предложения:

- «Jill and Mary are sisters.» — (они являются сёстрами по отношению друг к другу).
- «Jill and Mary are mothers.» — (каждая независимо является матерью).

Зависимость от поставленной задачи

В-пятых, постоянный задаче-независимый (task-independent) набор методов не имеет смысла, если учесть то, что многозначность слова мышь (животное и компьютерное устройство), например, вообще не влияет на результат англо-русского и русско-английского перевода (так как в обоих языках оба эти значения имеют воплощение в одном и том же слове), но сильно влияет при информационном поиске.

Дифференцированность значений слов

Актуальность проблемы, возможные применения

Информационный поиск

В системах поиска информации — если при поиске по запросу исключить из рассмотрения те документы, в которых какое-либо из слов запроса употребляется в не том значении, которое интересует пользователя в данный момент, то можно увеличить релевантность результатов запросов.

Машинный перевод

В системах **машинного перевода** отсутствие надежных механизмов распознавания значения слова значительно снижает качество

перевода, так как слово не всегда однозначно переводится на другой язык.

Извлечение информации

В специфичных областях наибольший интерес представляют проблемы разрешения специфичных им концептов: к примеру, в медицинской области может пригодиться определения названий лекарств в тексте, тогда как в биоинформатике необходимо разрешать неоднозначности в именовании генов и протеинов — этот процесс был назван **Извлечение информации** (Information Extraction).

Основные типы методов

- **Методы, основанные на знаниях**
- **Методы обучения с учителем**
- **Методы частичного обучения с учителем**
- **Методы обучения без учителя**
- **Другие методы**

Также существуют другие методы, основанные на совершенно отличающихся от вышеперечисленных принципах:

- Определение доминантности значения слова (Determining Word Sense Dominance)^{[30][31][32][33]}.
- Разрешение, основанное на темах (доменах) корпуса (Domain-Driven Disambiguation)^{[34][35]}
- WSD, использующее кросс-языковые данные (Cross-Lingual Evidence)

9. Автоматичне реферування текстів.

http://science.lp.edu.ua/sites/default/files/Papers/15_168.pdf

Автоматизоване реферування або квазіреферування — це виявлення в тексті первинного документа фрагментів, що містять заздалегідь заявлені змістові аспекти. Найвищого розвитку формалізація методів [реферування](#) набула з автоматизацією цього виду аналітико-синтетичної обробки документів. Необхідність реферування щораз більших обсягів документів і при цьому зменшення суб'єктивізму в наданні інформації зумовили впровадження в реферування електронних технологій.

Методи автоматизованого реферування

Методи автоматизованого реферування базуються на можливості виявлення в тексті первинного документа фрагментів, що містять заздалегідь заявлені змістові аспекти, і на формуванні з них рефератів-екстрактів. Фрагменти тексту первинного документа вибирають за формальними ознаками, а саме за частотою вживання слів, обраних як змістові критерії. На жаль, таких підхід не завжди гарантує відбір з тексту найважливіших відомостей, тому такі реферати виконують переважно пошукову і комунікативну функції. Щоб відрізнисти автоматичні реферати від інтелектуальних, перші часто називають квазірефератами, а процес автоматизованого реферування квазіреферуванням. Методи автоматизованого реферування поділяються на:

1. статистичні.
 2. позиційні.
 3. індикативні.

Суть **статистичних** методів, що засновані на статистичному аналізі текстів, — це методики російських вчених В. Аgraєва, Б. Бородіна та В.

Пурто. Перші двоє запропонували методику, згідно з якою вибрані з тексту речення виявляються пов'язаними між собою. Найбільш зв'язаними, а тому такими, що мають бути включені до реферату, вважаються речення, які містять найбільшу кількість однаково значущих слів. В. Пурто розробив метод оцінки та відбору речень за кількістю інформації, яку вони містять. У цьому випадку тексти підлягають статистичному аналізу для виявлення частоти вживання слів. Словами, що найчастіше вживаються у науково-технічній літературі, є терміни.

Позиційні методи вдосконалюють відбір найбільш значущих речень з текстів первинних документів з використанням складного математичного апарату.. Відбір здійснюється на засадах чотирьох взаємопов'язаних методів: натяку, ключових слів, заголовка, локалізації. Сутність методу натяку полягає у використанні під час відбору речень списку слів, в якому заздалегідь виділено слова з позитивною та негативною змістовою вагою, а також «нульові» (нейтральні) слова. При відборі враховуються тільки слова, що передають позитивну й негативну оцінку.

При використанні методу **ключових слів** розглядаються слова, відібрани за частотним принципом та за цією ознакою визначені ключовими, що є аналогічним до запропонованого Г. Луном підходу.

У методі **заголовка** головна роль відводиться словнику термінів, відібраних із заголовка та підзаголовків, які мають більшу «вагу», ніж слова з інших речень тексту. До реферату відбираються речення, де трапляються терміни, котрі наявні у словнику. Метод локалізації ґрунтується на припущення, що найсуттєвіша інформація концентрується на самому початку або наприкінці певного уривка чи параграфа тексту.

Індикативні методи дають змогу на основі синтаксичного аналізу формалізувати виклад основного змісту первинного документа в рефераті телеграфного стилю. Синтаксичному аналізу може підлягати як увесь текст, так і його окремі фрагменти, що містять типові маркери. Показником для виділення значущих елементів правлять розділові знаки в середині речення. Обсяг одержаних рефератів становить у середньому до 35 % обсягу першоджерела

10. Латентне розміщення Діріхле.

https://ru.wikipedia.org/wiki/Латентное_размещение_Дирихле

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

Метод латентного размещения Дирихле (latent Dirichlet allocation, LDA) предложен Дэвидом Блеем в 2003 году. В этом методе устранены основные недостатки PLSA.

Метод LDA основан на той же вероятностной модели

$$p(d, w) = \sum_{t \in T} p(d)p(w|t)p(t|d),$$

при дополнительных предположениях:

- векторы документов $\theta_d = (p(t|d): t \in T)$ порождаются одним и тем же вероятностным распределением на нормированных $|T|$ -мерных векторах; это распределение удобно взять из параметрического семейства распределений Дирихле $\text{Dir}(\theta, \alpha)$, $\alpha \in \mathbb{R}^{|T|}$;
 - векторы тем $\phi_t = (p(w|t): w \in W)$ порождаются одним и тем же вероятностным распределением на нормированных векторах размерности $|W|$; это распределение удобно взять из параметрического семейства распределений Дирихле $\text{Dir}(\theta, \beta)$, $\beta \in \mathbb{R}^{|W|}$.

Для идентификации параметров модели LDA по коллекции документов применяется [самплирование Гиббса](#), вариационный байесовский вывод или метод [Expectation-Propagation](#).

Латентное размещение Дирихле (*LDA*, от англ. *Latent Dirichlet allocation*) — применяемая в [машинном обучении](#) и [информационном поиске](#) **порождающая модель**, позволяющая объяснять результаты наблюдений с помощью [неявных](#) групп, благодаря чему возможно выявление причин сходства некоторых частей данных. Например, если наблюдениями являются слова, собранные в документы, утверждается, что каждый документ представляет собой смесь небольшого количества тем и что появление каждого слова связано с одной из тем документа

В LDA каждый документ может рассматриваться как набор различных тематик. Подобный подход схож с [вероятностным латентно-семантическим анализом](#) (pLSA) с той разницей, что в LDA предполагается, что распределение тематик имеет в качестве [априори распределения Дирихле](#). На практике в результате получается более корректный набор тематик.

К примеру, модель может иметь тематики классифицируемые как «относящиеся к кошкам» и «относящиеся к собакам», тематика обладает вероятностями генерировать различные слова, такие как «мяу», «молоко» или «котёнок», которые можно было бы классифицировать как «относящиеся к кошкам», а слова, не обладающие особой значимостью (к примеру, [служебные слова](#)), будут обладать примерно равной вероятностью в различных тематиках.

11. Моделі векторизації семантики слів.

<https://towardsdatascience.com/https-medium-com-tanaygahlot-moving-beyond-the-distributional-model-for-word-representation-b0823f1769f8>

<https://pdfs.semanticscholar.org/2109/bf2265bea1933667da9baf207e13cc7fb016.pdf>

Теперь мы представляем модель, которая изучает представления слов из информации о терминологическом документе, используя принципы, аналогичные тем, которые используются в `lBLM` и других моделях нейронного языка. Однако, в отличие от предыдущей работы в литературе по модели нейронного языка, наша модель естественным образом обрабатывает данные терминов-документов для изучения векторов семантических слов. Мы выводим вероятностную модель с лог-билинейной энергетической функцией для моделирования пакета слов в документе. Этот подход, естественно, обрабатывает длинные документы переменной длины и изучает представления, чувствительные к корреляциям слов на большом расстоянии. Обучение с максимальным правдоподобием может быть эффективно выполнено с помощью оптимизации подъема по координатам.

Распределительные модели страдают от следующей проблемы:

Редкие слова: они не изучают хорошее представление для слов с более низкой частотой встречаемости в корпусе.

Смешение смысла: они объединяют все смыслы слова в одно, например, слово «банк» может относиться к «берегу реки» или «финансовому учреждению». Распределительная модель объединяет эти интерпретации в одну.

Отсутствует морфология: они не принимают во внимание морфологию слова при обучении представлению. Например, слова «оценивать» и

«оценивает» имеют сходное значение, но они рассматриваются как два отдельных слова.

В общих чертах эти подходы делятся на **3 основных класса**, мы рассмотрим их в следующем порядке:

- Морфология чувствительного встраивания
- Увеличение лингвистики или функциональных ограничений в вложения слов
- Обработка смыслов этого слова.

Морфология чувствительного встраивания

Этот класс методов учитывает морфологию слова при изучении вложений. Fasttext - примерный пример этого класса техники. Это слово трактуется как сумма представления n-грамм символов. Например, слово « where» представляется как “<wh, whe, her, ere, re>”. Каждому символу n-граммы назначается вектор, который впоследствии используется для вычисления оценки соответствия между вектором контекста и вектором назначения:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

fasttext может обеспечить вложение для слова, которое никогда не встречалось в корпусе, поскольку оно представляет слово как сумму известного символа n-граммы.

Увеличение лингвистики или функциональных ограничений в вложения слов

Другой класс методов для специализации пространства слов - постобработка встраивания слов с лингвистическими / функциональными ограничениями.

Attract-Repel - это метод постобработки, который берет предварительно обученное вложение и специализирует его в

соответствии с лингвистическими ограничениями. Например, в морффитинге языковые ограничения были выражены в виде двух множеств

English	German	Italian
(discuss, discussed)	(schottisch, schottischem)	(golfo, golfi)
(laugh, laughing)	(damalige, damaligen)	(minato, minata)
(pacifist, pacifists)	(kombiniere, kombinierte)	(mettere, metto)
(evacuate, evacuated)	(schweigt, schweigst)	(crescono, cresci)
(evaluate, evaluates)	(hacken, gehackt)	(crediti, credite)
<hr/>		
(dressed, undressed)	(stabil, unstabil)	(abitata, inabitato)
(similar, dissimilar)	(geformtes, ungeformt)	(realtà, irrealità)
(formality, informality)	(relevant, irrelevant)	(attuato, inattuato)

Table 2: Example synonymous (inflectional; top) and antonymous (derivational; bottom) constraints.

Верхняя половина таблицы показывает набор притяжения, а нижняя половина матрицы показывает набор отталкивания. Используя эти наборы, формируется мини-партия, которая используется для оптимизации следующей функции потерь:

$$C(\mathcal{B}_S, \mathcal{B}_A) = S(\mathcal{B}_S) + A(\mathcal{B}_A) + R(\mathcal{B}_S, \mathcal{B}_A)$$

Первое слагаемое в этой функции потерь соответствует множеству притяжения, а второе слагаемое соответствует набору отталкивания. Третий член сохраняет дистрибутивное представление. Кроме того, первые два термина также прививают отрицательные примеры - идея, заимствованная из модели PARAGRAM. Функция стоимости для первых двух слагаемых определяется как:

$$\begin{aligned} A(\mathcal{B}_A) = \sum_{(x_l, x_r) \in \mathcal{B}_A} & [\tau (\delta_{ant} + \mathbf{x}_l \mathbf{x}_r - \mathbf{x}_l \mathbf{t}_l) \\ & + \tau (\delta_{ant} + \mathbf{x}_l \mathbf{x}_r - \mathbf{x}_r \mathbf{t}_r)] \end{aligned}$$

Обработка смыслов слова

В ELMO слово векторизовано на основе контекста. Следовательно, для векторизации слова также необходимо указать контекст, в котором оно встречается. Этот подход оказался действительно эффективным по сравнению с теми методами векторизации, которые не учитывают контекст. Пример для того же самого можно увидеть, когда сравнивают ближайшего соседа из ELMO (biLM) и Glove:

Основная идея ELMO состоит в том, чтобы генерировать внедрение как взвешенную сумму внутреннего состояния уровней модели двунаправленного языка и представления окончательного уровня сети свертки символов.

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \dots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \dots, L\}, \end{aligned}$$

$$\text{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^L s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

12. Розпізнавання іменованих сутностей.

[http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=Progr_2016_2-3\(spets._18](http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=Progr_2016_2-3(spets._18)

Основною задачею системи є розпізнавання у тексті іменованих сутностей та визначення типу цих сутностей. Вхідними даними системи є текст, написаний правильною англійською мовою з мінімальним вживанням сленгу та відсутністю орфографічних і граматичних помилок.

Архітектурно система складається з кількох ключових блоків, кожен блок виконує функції певного етапу побудови розв'язку задачі. Усі модулі попередньої обробки тексту для перетворення його у необхідний системі вигляд внесено за межі системи.

Система структурно складається з наступних блоків:

- ❑ блок ідентифікації та аналізу іменованих сутностей на основі Байесівської моделі;
- ❑ блок ідентифікації та аналізу іменованих сутностей на основі моделі умовних випадкових полів – Conditional random field (CRF).

Всі блоки є підсистемами, які паралельно і незалежно одна від одної виконують наступну обробку вхідного тексту:

- ідентифікація синтаксичних груп речень тексту, які містять іменовані сутності;
- визначення меж знайдених іменованих сутностей (перше слово сутності – останнє слово сутності);
- визначення типів знайдених іменованих сутностей.

Підсистеми виконують дану обробку тексту з відповідною розміткою. Результатом роботи системи є текст з відповідною розміткою іменованих сущностей (id сущності, границі сущності, тип сущності). Система налаштована для розпізнавання наступних типів іменованих сущностей (Type in system), кожен тип трактується у відповідності до його трактування

Вхідними даними для розроблених класифікаторів є текст англійською мовою, дерева виведення та за- лежностей речень вхідного тексту, а також всі дані стосовно лексичних значень слів речень тексту згідно розмітки

13. Embeddings моделі представлення слів та словосполучень.

<https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346>

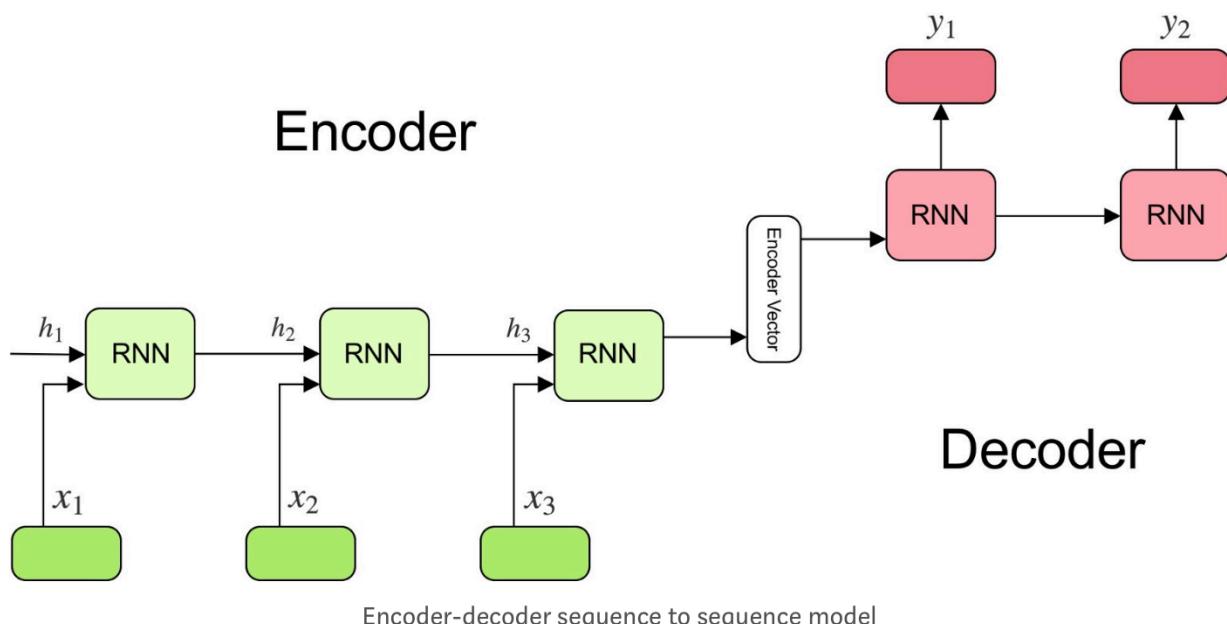
<https://towardsdatascience.com/training-and-visualising-word-vectors-2f946c6430f8>

<https://towardsdatascience.com/3-silver-bullets-of-word-embedding-in-nlp-10fa8f50cc5a>

<https://habr.com/ru/company/ods/blog/329410/>

Definition of the Sequence to Sequence Model

Вперше представлена Google в 2014 году, модель «Sequence to Sequence Model» предназначена для отображения входных данных фиксированной длины с выходными данными фиксированной длины, где длина входных и выходных данных может различаться.



Модель состоит из 3 частей: кодер, промежуточный (кодер) вектор и декодер.

кодировщик (Encoder)

- Стек из нескольких повторяющихся блоков (ячеек LSTM или GRU для лучшей производительности), где каждый принимает один элемент входной последовательности, собирает информацию для этого элемента и распространяет ее вперед.
- В задаче с ответом на вопрос входная последовательность представляет собой набор всех слов из вопроса. Каждое слово представлено как x_i , где i - порядок этого слова.
- Скрытые состояния h_i вычисляются по формуле:

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

Эта простая формула представляет собой результат обычной рекуррентной нейронной сети. Как видите, мы просто применяем соответствующие веса к предыдущему скрытому состоянию $h_{(t-1)}$ и входному вектору x_t .

Кодер вектор (Encoder Vector)

- Это окончательное скрытое состояние, создаваемое частью кодировщика модели. Он рассчитывается по формуле выше.
- Этот вектор стремится инкапсулировать информацию для всех входных элементов, чтобы помочь декодеру делать точные прогнозы.
- Он действует как начальное скрытое состояние части модели декодера.

Decoder

- Стек из нескольких повторяющихся блоков, каждый из которых предсказывает выходной сигнал y_t на временном шаге t .

- Каждый рекуррентный модуль принимает скрытое состояние от предыдущего модуля и производит и выводит, а также свое собственное скрытое состояние.
- В задаче с ответом на вопрос выходная последовательность представляет собой совокупность всех слов из ответа. Каждое слово представлено как y_i , где i - порядок этого слова.
- Любое скрытое состояние h_i вычисляется по формуле:

$$h_t = f(W^{(hh)} h_{t-1})$$

Как видите, мы просто используем предыдущее скрытое состояние для вычисления следующего.

- Выходные данные y_t на временном шаге t вычисляются по формуле:

$$y_t = \text{softmax}(W^S h_t)$$

Мы рассчитываем выходные данные, используя скрытое состояние на текущем временном шаге вместе с соответствующим весом W (S). Softmax используется для создания вектора вероятности, который поможет нам определить конечный результат (например, слово в задаче ответа на вопрос).

Сила этой модели заключается в том, что она может отображать последовательности различной длины друг на друга. Как видите, входы и выходы не коррелированы и их длина может отличаться. Это открывает целый ряд новых проблем, которые теперь можно решить с помощью такой архитектуры.

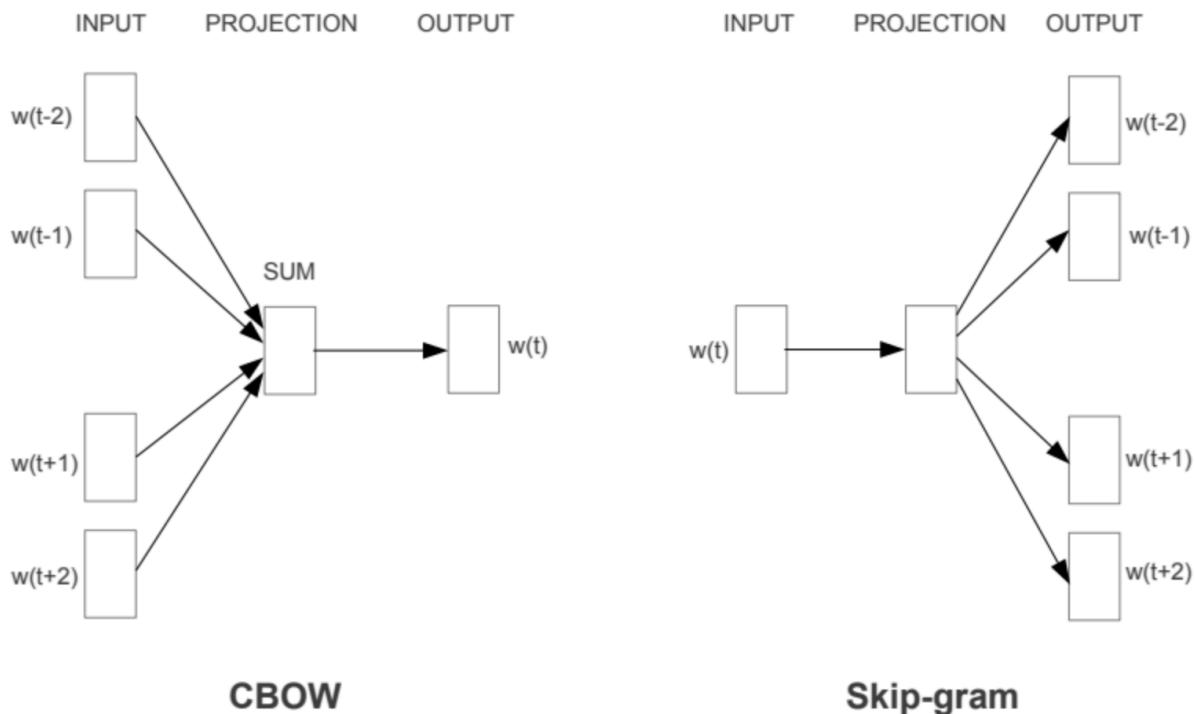
Continuous bag-of-words (CBOW) & Skip-gram

CBOW is that using both **n** words before and after target word (**w**). For instance, “the word vector encodes semantic relationship among words”. If the window (n) is 3, here is the subset of prediction list:

- **Case 1, Before Words:** {Empty}, After Words: (word, vector, encodes), Predict Word: “the”
- **Case 2, Before Words:** (the), After Words: (vector, encodes semantic), Predict Word: “word”

Skipgram использует противоположный подход, который использует целевое слово, чтобы предсказать n слов до и после целевого слова. Например, «вектор слова кодирует семантические отношения между словами». Если окно (n) равно 3, здесь есть подмножество списка предсказаний:

- Case 1, Predict Word: “the”, Words: (word, vector, encodes)
- Case 2, Predict Word: “word”, Words: (the, vector, encodes, semantic)



Negative Sampling

Negative Sampling

В стандартной модели CBOW, рассмотренной выше, мы предсказываем вероятности слов и оптимизируем их. Функцией для оптимизации (минимизации в нашем случае) служит дивергенция Кульбака-Лейблера:

$$KL(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Здесь $p(x)$ — распределение вероятностей слов, которое мы берем из корпуса, $q(x)$ — распределение, которое порождает наша модель. Дивергенция — это буквально "расхождение", насколько одно распределение не похоже на другое. Т.к. наши распределения на словах, т.е. являются дискретными, мы можем заменить в этой формуле интеграл на сумму:

$$KL(p||q) = \sum_{x \in V} p(x) \log \frac{p(x)}{q(x)}$$

Оказалось так, что оптимизировать эту формулу достаточно сложно. Прежде всего из-за того, что $q(x)$ рассчитывается с помощью softmax по всему словарю. (Как мы помним, в английском сейчас порядка миллиона слов.) Здесь стоит отметить, что многие слова вместе не встречаются, как мы уже отмечали выше, поэтому большая часть вычислений в softmax является избыточной. Был предложен элегантный обходной путь, который получил название Negative Sampling. Суть этого подхода заключается в том, что мы максимизируем вероятность встречи для нужного слова в типичном контексте (том, который часто встречается в нашем корпусе) и одновременно минимизируем вероятность встречи в нетипичном контексте (том, который редко или вообще не встречается).

Формулой мысль выше записывается так:

$$NegS(w_o) = \sum_{i=1, x_i \sim D}^{i=k} -\log(1 + e^{s(x_i, w_o)}) + \sum_{j=1, x_j \sim D'}^{j=l} -\log(1 + e^{-s(x_j, w_o)})$$

Здесь $s(x, w)$ — точно такой же, что и в оригинальной формуле, а вот остальное несколько отличается. Прежде всего стоит обратить внимание на то, что формуле теперь состоит из двух частей: позитивной ($s(x, w)$) и негативной ($-s(x, w)$). Позитивная часть отвечает за типичные контексты, и D здесь — это распределение совместной встречаемости слова w и остальных слов корпуса. Негативная часть — это, пожалуй, самое интересное — это набор слов, которые с нашим целевым словом встречаются редко. Этот набор порождается из распределения D' , которое на практике берется как равномерное по всем словам словаря корпуса. Было показано, что такая функция приводит при своей оптимизации к результату, аналогичному стандартному softmax [2].

Word2Vec

Это векторное слово обучено в новостях Google и предоставлено Google.

GloVe

Глобальные векторы для представления слов (GloVe) предоставляются командой Stanford NLP. Стэнфорд предлагает различные модели от 25, 50, 100, 200 до 300 измерений на основе 2, 6, 42, 840 миллиардов токенов.

Команда Stanford NLP применяет вероятность совпадения слово-слово для построения вложения. Другими словами, если два слова сосуществуют много раз, оба слова могут иметь одинаковое значение, поэтому матрица будет ближе.

fastText

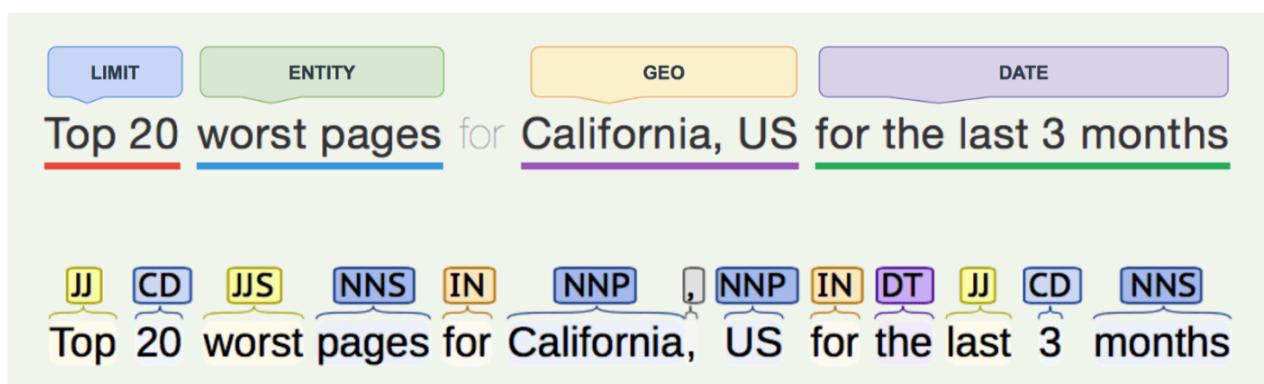
FastText выпущен Facebook, который предоставляет 3 модели с 300 размерами. Одна из предварительно обученных моделей обучается с подсловом. For example, “difference”, it will be trained by “di”, “dif”, “diff” and so on.

14. Моделі представлення семантики речень.

<https://chatbotslife.com/introduction-into-semantic-modelling-for-natural-language-processing-c4c885175ca7>

Семантическая и лингвистическая грамматики определяют формальный способ понимания предложения на естественном языке. Лингвистическая грамматика имеет дело с лингвистическими категориями, такими как существительное, глагол и т. д. С другой стороны, семантическая грамматика - это тип грамматики, терминами которой являются не общие структурные или лингвистические категории, такие как существительные или глаголы, а скорее семантические категории, такие как PERSON или COMPANY.

В свою очередь, семантическое моделирование вначале имело начальный всплеск интереса, но быстро сошло на нет из-за технических сложностей. Однако в последние годы семантическое моделирование пережило ренессанс, и теперь оно является основой практически всех коммерческих систем НЛП, таких как Google, Cortana, Siri, Alexa и т. д. Это также лежит в основе DataLingvo (компания, в которой я работаю где мы продолжили развивать идею семантического моделирования).



На рисунке выше нижнее и верхнее предложения одинаковы, но они обрабатываются по-разному. Нижняя часть анализируется с использованием традиционной лингвистической грамматики, где каждое слово помечается тегом PoS (Point-of-Speech), например, NN для nouns, JJ для прилагательного и так далее. Однако верхняя часть

анализируется с использованием семантической грамматики, и вместо того, чтобы отдельные слова были помечены PoS, одно или несколько слов образуют семантические категории высокого уровня, такие как DATE или GEO.

Пример семантической грамматики

Независимо от конкретного синтаксиса конфигурации грамматика обычно определяется как набор семантических объектов, где каждый объект как минимум имеет имя и список синонимов, по которым этот объект может быть распознан.

```
<WEBSITE>:  
    website,  
    http website,  
    https website,  
    http domain,  
    web address,  
    online address,  
    http address  
<USER>:  
    user,  
    web user,  
    http user,  
    https user,  
    online user
```

Determinism

Причина этого в природе самой семантической грамматики, которая основана на простом сопоставлении синонимов. Правильно определенная семантическая грамматика обеспечивает полностью детерминированный поиск семантического объекта. Буквально нет «догадок» - семантическая сущность либо однозначно найдена, либо нет.

Семантическая грамматика с сопоставлением на основе синонимов ограничивается конкретной, часто очень узкой, областью данных. Причиной этого является тот факт, что для создания семантической модели необходимо создать исчерпывающий набор всех сущностей и, что самое сложное, набор всех их синонимов.

Семантическая модель с базовым набором синонимов для ваших семантических сущностей, которые можно сделать довольно быстро. Как только приложение NLP / NLU, использующее эту модель, начинает работать, пользовательские предложения, которые не могут быть автоматически «поняты» этой моделью, перейдут к курированию. Во время человеческого курирования пользовательское предложение будет изменено, чтобы соответствовать модели, и алгоритм самообучения «усвоит» эту поправку и выполнит ее автоматически в следующий раз, не требуя человеческой передачи.

В этом процессе есть два критических свойства:

- Человеческое курирование изменяет вводимые пользователем данные в соответствии с существующей текущей семантической моделью, то есть пользовательское предложение изменяется таким образом, что на него можно автоматически отвечать. Как правило, это включает исправление орфографических ошибок, разговорный язык, сленг, удаление стоп-слов или добавление недостающего контекста.
- Это изменение (то есть курирование) в предложении пользователя вводится в алгоритм самообучения, который «запоминается» на будущее. Так как это изменение было первоначально выполнено человеком, который делает самообучение контролируемым процессом и устраняет введение кумулятивных ошибок в обучении.

15. Задача визначення авторського стилю. Моделі та методи.

<https://ru.wikipedia.org/wiki/>

[%D0%90%D0%B2%D1%82%D0%BE%D1%80%D1%81%D0%BA%D0%B8%D0%B9 %D0%B8%D0%BD%D0%B2%D0%B0%D1%80%D0%B8%D0%BD%D1%82](#)

<https://bit.ly/2L08gPH>

Авторский инвариант ([англ. writer invariant, authorial invariant, author's invariant](#)) — это количественная характеристика литературных текстов или некий параметр, который однозначно характеризует своим поведением произведения одного автора или небольшого числа «близких авторов», и принимает существенно разные значения для произведений разных групп авторов.[источник не указан 715 дней] Авторский инвариант применяется в задаче идентификации авторства текста.[⇨]

Задача идентификации авторства текста — это задача установления авторства неизвестного текста с помощью выделения особенностей авторского стиля и сравнения этих особенностей с другими произведениями, авторство которых известно.

Идентификация авторства текста [[править](#) | [править код](#)]

Задача идентификации авторства текста [[править](#) | [править код](#)]

Формулировка задачи идентификации автора текста при ограниченном наборе альтернатив выглядит следующим образом:^[1]

$T = \{t_1, \dots, t_k\}$ – множество текстов,

$A = \{a_1, \dots, a_l\}$ – множество авторов.

Для некоторого подмножества текстов $T' = \{t_1, \dots, t_m\} \subseteq T$ авторы известны, т.е. существует множество пар «текст–автор» $D = \{(t_i, a_j)\}_{i=1}^m$. Необходимо установить, кто из множества A является истинным автором остальных текстов (анонимных или спорных) $T'' = \{t_{m+1}, \dots, t_k\} \subseteq T$

Методика идентификации автора неизвестного текста [[править](#) | [править код](#)]

Методика включает последовательность следующих действий:^[1]

1. Выбор модели представления текстов в виде наборов признаков.
2. Выбор группы признаков для проверки и формирования из неё авторского инварианта.
3. Выбор классификаторов и их параметров.
4. Формирование модели авторского стиля, позволяющей разделять двух и более авторов на основе полученного авторского инварианта и обученного классификатора.
5. Непосредственно определение авторства неизвестного текста.
6. Принятие итогового решения об авторе текста ансамблем классификаторов в случае, если удалось найти несколько информативных групп признаков текста.

Описание [[править](#) | [править код](#)]

Основные свойства, которыми должна обладать числовая характеристика авторского инварианта:

1. Она должна быть достаточно «массовой», интегральной, чтобы слабо контролироваться автором на сознательном уровне. Другими словами, она должна быть его «бессознательным параметром», коренящимся настолько глубоко, что автор даже не задумывается о нем. А если бы даже задумался, то не смог бы долго его контролировать и в результате довольно быстро вернулся бы в прежнее устойчивое и типичное для него состояние.
2. Искомый параметр должен сохранять «постоянное значение» для произведений данного автора. То есть, иметь небольшое отклонение от среднего значения (слабо колебаться) на протяжении всех его книг. Именно это свойство и позволяет говорить, что данный параметр является инвариантом.
3. Параметр должен уверенно различать между собой разные группы писателей. Другими словами, должно существовать достаточное число авторских групп, заметно отличающихся друг от друга значениями инварианта.^[источник не указан 715 дней]

Такими количественными характеристиками могут быть:

1. Длина предложений, то есть среднее число слов в предложении.
2. Длина слов, то есть среднее количество слогов в слове.
3. Общая частота употребления служебных слов - предлогов, союзов, частиц, то есть процентное содержание служебных слов.
4. Частота употребления существительных, то есть их процентное содержание.
5. Частота употребления глаголов, то есть их процентное содержание.
6. Частота употребления прилагательных (в процентах).
7. Частота употребления предлога «в» (в процентах).
8. Частота употребления частицы «не» (в процентах).
9. Количество служебных слов в предложении, то есть среднее число союзов, предлогов и частиц в предложении.^[источник не указан 715 дней]

2. МЕТОД СРАВНЕНИЯ СТИЛЕЙ ТЕКСТОВ

2.1. Двухвыборочный критерий

Двухвыборочные критерии применяются в задаче проверки гипотезы о принадлежности двух независимых выборок одному закону распределения в евклидовом пространстве \mathbf{R}^d .

Кижаева Н.А., Шалымов Д.С.

Пусть $X = X_1, X_2, \dots, X_m$ и $Y = Y_1, Y_2, \dots, Y_n$ — две независимые случайные величины с неизвестными функциями распределения F и G . Задача состоит в проверке гипотезы:

$$H_0 : F(x) = G(x).$$

Альтернативная гипотеза:

$$H_1 : F(x) \neq G(x).$$

Критерий типа Колмогорова-Смирнова — один из самых распространенных непараметрических критериев проверки однородности двух выборок.

Статистика критерия для эмпирических функций распределений $\tilde{F}(x)$ и $\tilde{G}(x)$ определяется следующим образом:

$$D = \sup_x |\tilde{F}(x) - \tilde{G}(x)|$$

Асимптотический критерий не зависит от распределения, поэтому для достаточно больших выборок тестовая статистика не зависит от их исходных распределений. Он также применим и к задаче проверки соответствия выборки некоторому закону распределения (так называемый критерий согласия Колмогорова). В этом случае статистика измеряет расстояние между эмпирической функцией распределения выборки и предполагаемым кумулятивным распределением.

В работах [12] и [13] представлен обзор непараметрических критериев для многомерного случая, дан их сравнительный анализ. Обобщение критерия Смирнова в многомерном случае дано в статье [14].

Статистика двухвыборочного критерия призвана описывать качество «смешения» элементов, принадлежащих двум независимым одинаково распределенным выборкам S_1 и S_2 . Это качество можно измерить с помощью отношений K ближайших соседей, посчитанных для каждого элемента выборок. Если выборки хорошо «замешаны», то такие пропорции приблизительно одинаковые.

Обозначим $|\cdot|$ — произвольная норма в пространстве R^d и положим

$$Z_i = \begin{cases} X_i & 1 \leq i \leq m, \\ Y_{i-m} & m+1 \leq i \leq l, \end{cases}$$

где $l = m + n$ — размер выборки. Ближайший r -й сосед для Z_i — это такой элемент Z_j , что $|Z_v - Z_i| < |Z_j - Z_i|$ ровно для $r-1$ значений v , $1 < v < l$, $v \neq i, j$.

В статье рассматривается следующая статистика:

$$T_K(S_1 \cup S_2) = \sum_{x \in S_1 \cup S_2} \sum_{r=1}^K I\left(\begin{array}{c} x \text{ и } r\text{-й сосед} \\ \text{принадл. одной выборке} \end{array} \right)$$

которая представляет все совпадения с K ближайшими соседями.

В условиях гипотезы о равенстве распределений в объединенной выборке в среднем меньше совпадений с ближайшими соседями, чем при альтернативной, поэтому по критерию отвергается нулевая гипотеза для больших значений статистики T_K .

Доказано, что предельное распределение такого рода статистик — нормальное для произвольной нормы в пространстве R^d [15]. При сравнении двух реальных текстов из-за их неоднородности невозможно использовать асимптотически нормальное распределение. В то же время распределение для нулевой гипотезы можно смоделировать в духе бутстрата (англ. *bootstrap*) [16]. Построение эмпирической функции распределения объединенных выборок подразумевает равенство теоретических распределений согласно

нулевой гипотезе. В то же время в случае разных законов распределения выше описанная процедура (использование только «первоначального смешивания») может выдать зашумленное распределение. Поэтому ниже приводится точное описание процесса генерации выборок.

2.2. Алгоритм

Для начала рассматриваемые тексты преобразуются в бинарные файлы F_1 , F_2 и вводится $F_0 = F_1 \cup F_2$. Важной частью алгоритма, нацеленного на различие распределений этих файлов, является процедура создания повторной выборки (англ. *re-sampling*). Выборки формируются с помощью N -грамм, последовательностей N символов из текста, как показано в Алгоритме 1.

Algorithm 1 Процедура сэмплирования

Require:

- F — текстовый файл;
- N — размер атрибута (N -граммы);
- $NWORD$ — количество атрибутов в векторе (размерность вектора);
- $NVEC$ — количество векторов в выборке (размер выборки).

repeat $NVEC$ раз

1. Сгенерировать случайное число — начальную позицию вектора в файле;
 2. Построить вектор из $NWORD$ последовательных атрибутов.
-

16. Модель Glove.

<https://nlp.stanford.edu/projects/glove/>

<https://towardsdatascience.com/emnlp-what-is-glove-part-i-3b6ce6a7f970>

<https://towardsdatascience.com/emnlp-what-is-glove-part-ii-9e5ad227ee0>

<https://nlp.stanford.edu/pubs/glove.pdf>

Мы не будем полностью рассматривать математику разложения по сингулярным значениям в этой статье, но по сути это факторизация общей матрицы m -by- n матрицы M в произведение $U \Sigma V^*$, где U m -by- m и унитарное, Σ - прямоугольная диагональная матрица размером m на n (ненулевые элементы которой известны как сингулярные значения M), а V - n -на- n и унитарная.

$$\begin{array}{cccc} \text{Matrix } M & = & \text{Matrix } U & \text{Matrix } \Sigma & \text{Matrix } V^* \\ m \times n & & m \times m & m \times n & n \times n \\ \\ \text{Matrix } U & & \text{Matrix } U^* & = & \text{Matrix } I_m \\ \\ \text{Matrix } V & & \text{Matrix } V^* & = & \text{Matrix } I_n \end{array}$$

Diagram illustrating the decomposition of matrix M into $U \Sigma V^*$. The matrices are shown as grids:

- M : A 4×5 grid of grey squares.
- U : A 4×4 grid with columns colored blue, green, blue, green.
- Σ : A 4×5 grid with a yellow square at position (4,1) and a blue square at position (4,2).
- V^* : A 5×4 grid with rows colored purple, pink, purple, pink.
- U^* : A 4×4 grid with columns colored blue, green, blue, green.
- I_m : A 4×4 identity matrix with ones on the diagonal.
- V : A 5×4 grid with rows colored purple, pink, purple, pink.
- I_n : A 4×4 identity matrix with ones on the diagonal.

Напомним, что сопряженная транспонирование A^* матрицы A - это матрица, заданная путем взятия комплексного сопряжения каждой

записи в транспонировании (отражение по диагонали) в А. Унитарной матрицей является любая квадратная матрица, у которой сопряженная транспонирование является ее обратной, т.е. матрица А такая, что $AA^* = A^*A = I$. Эта факторизация затем используется для нахождения аппроксимации низкого ранга для М, сначала выбирая r , желаемый ранг нашей матрицы аппроксимации M' , а затем вычисляя Σ' , который это просто Σ , но только с r самыми большими сингулярными значениями. Тогда наше приближение дается формулой $M' = U\Sigma'V^*$.

Эти аппроксимации низкого ранга для термина частотных матриц дают нам вложения разумного размера в векторное пространство глобальной статистики корпуса.

Окно локального контекста.

Другая модель встраивания слов изучает семантику, пропуская окно над корпусом построчно и учась предсказывать либо окружение данного слова (модель скрап-грамммы), либо предсказывать слово, исходя из его окружения (непрерывный пакет- модель слов). Обратите внимание, что проблема с пакетом слов часто сокращается до «CBOW».

Вероятности совпадения.

Вспомните из первой части этой серии, что частотные матрицы терминов кодируют частоту появления терминов в контексте друг друга, перечисляя каждый уникальный токен в корпусе вдоль обеих осей большой двумерной матрицы. Выполнение факторизации матрицы дает нам низкое ранговое приближение ко всем данным, содержащимся в исходной матрице. Однако, как мы вскоре объясним, авторы GloVe обнаружили с помощью эмпирических методов, что вместо изучения необработанных вероятностей совместного появления, возможно, имеет больше смысла изучать отношения этих вероятностей совместного появления, которые, кажется, лучше различают тонкости в срочности.

Чтобы проиллюстрировать это, мы заимствуем пример из их статьи: предположим, мы хотим изучить отношения между двумя словами: $i =$ лед и $j =$ пар. Мы сделаем это путем изучения вероятностей совпадения этих слов с различными «пробными» словами. Мы определяем вероятность совпадения произвольного слова i с

произвольным словом j как вероятность появления слова j в контексте слова i . Это представлено уравнением и определениями ниже.

CO-OCCURRENCE PROBABILITY

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_k X_{ik}},$$

X_{ij} = number of times word j
occurs in the context of word i .

Примечание. X_i определяется как число раз, когда любое слово встречается в контексте слова i , поэтому оно определяется как сумма по всем словам k числа случаев, когда слово k встречается в контексте слова i . Поэтому, если мы выберем пробное слово $k = \text{solid}$, которое тесно связано с $i = \text{ice}$, но не с $j = \text{steam}$, мы ожидаем, что отношение вероятностей совместного появления $P_{\{ik\}} / P_{\{jk\}}$ будет большим, поскольку solid должно в теории, появляться в контексте льда чаще, чем в контексте пара, поскольку лед - это твердое тело, а пара - нет. Наоборот, для выбора $k = \text{газ}$ мы ожидаем, что такое же соотношение будет маленький, так как пар более тесно связан с газом, чем лед. Тогда у нас также есть такие слова, как вода, которые тесно связаны как со льдом, так и с паром, но не более одного, чем другого. И у нас также есть такие слова, как мода, которые не тесно связаны ни с одним из рассматриваемых слов. Как для воды, так и для моды, мы ожидаем, что наше соотношение будет близко к 1, поскольку не должно быть никакого смещения в отношении льда или пара.

Теперь важно отметить, что, поскольку мы пытаемся определить информацию о связи между словами «лед» и «пар», вода не дает нам много полезной информации. В целях дискриминации это не дает нам четкого представления о том, насколько «далеко друг от друга» пар находится от льда, а информация о том, что пар, лед и вода связаны между собой, уже зафиксирована в различающей информации между льдом и водой, и пар и вода. Слова, которые не помогают нам различать i и j , называются точками шума, и использование отношения

вероятностей совместного появления помогает отфильтровывать эти точки шума. Это хорошо иллюстрируется реальными данными для этих примеров, которые мы печатаем здесь из Таблицы 1 в статье GloVe.

NAIVE MODEL

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}.$$

Обратите внимание, что w являются действительными векторами слов. Теперь, так как мы хотим закодировать информацию о соотношениях между двумя словами, авторы предлагают использовать векторные различия в качестве входных данных для нашей функции. Тогда у нас есть следующее:

VECTOR DIFFERENCE MODEL

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}.$$

$$J = \sum_{i,j=1}^V f(X_{ij})(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik})^2,$$

где f - наша весовая функция, которая требует следующих свойств:

- $f(0) = 0$. Если предполагается, что f непрерывен, мы хотим, чтобы он достаточно быстро шел к нулю, чтобы $f(x) \log^2(x)$ имел конечный предел, так как в противном случае для совпадений, равных нулю, мы Функция может захотеть «взорвать» величины скалярного произведения и смещения, чтобы компенсировать лог.

- $f(x)$ должно быть неубывающим. Это относится к проблеме, которую мы изложили в параграфе выше. Это гарантирует, что небольшие записи в матрице не перевешиваются, и весит больше, чем больше. Авторы отмечают, что не исключено, что 85% записей X равны нулю.
- $f(x)$ должен быть относительно небольшим для больших значений x , то есть он не взрывается слишком быстро. Все, что быстрее x^2 , вероятно, не сработает. Я прошу прощения за использование здесь «быстрее» для описания функций более высокой степени, когда вы, сотрудники CS, вероятно, привыкли думать медленнее, как в контексте сложности времени, но, поскольку мы имеем дело с этими функциями с аналитической точки зрения, я думаю, что это делает больше смысл. Это просто для того, чтобы мы не переусердствовали с частыми совместными случаями.

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} .$$

Мы определяем x_{\max} как некоторую фиксированную константу, которая может быть больше или меньше, чем истинная наибольшая запись X . Мы используем второй случай, чтобы гарантировать, что функция все еще нормализована, даже если в X есть записи больше x_{\max} . Авторы обнаружили, что для альфа значение 3/4 дало немного лучшую производительность, чем интуитивное значение 1.

17. Information Retrieval.

<https://uk.wikipedia.org/wiki/>

%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%
86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9%D0%BF%D0%BE%D
1%88%D1%83%D0%BA

Інформаційний пошук (ІП) — наука про пошук неструктурованої документальної інформації. Особливо це відноситься до пошуку інформації в документах, пошук самих документів, добуття метаданих з документів, пошуку тексту, зображень, відео та звуку у локальних реляційних базах даних, у гіпертекстових базах даних таких, як Інтернет та локальні інtranет.

Автоматичні системи інформаційного пошуку використовують для зменшення так званого «інформаційного перевантаження». Багато університетів та публічних бібліотек використовують системи ІП для полегшення доступу до книжок, журналів та інших документів. Найвідомішим прикладом систем ІП можна назвати пошукові системи в Інтернеті.

Об'єктом інформаційного пошуку є текстова інформація, зображення, аудіо, відео інформація.

Існують такі стратегії інформаційного пошуку:

- з використанням векторно-просторового представлення (vector space model);
- пошук імовірності появи пошукового терміна в документі (probabilistic retrieval);
- з побудовою мовної моделі для кожного документа (language models);
- з побудовою мережі припущень, яка використовується для встановлення відповідності документу до пошукового запиту (inference network);
- з **Булевим** індексуванням, коли кожному пошуковому терміну присвоюється своя «вага», що потім враховується при побудові впорядкованих списків документів (Boolean indexing);

- з використанням не прояведеного семантичного індексування (latent semantic indexing);
- з побудовою **нейромереж** (neural networks);
- з використанням продуктивних алгоритмів, коли початковий пошуковий запит «еволюційно» видозмінюється (genetic algorithms);
- з використанням **нечітких множин**, коли документу ставиться у відповідність нечітка множина (fuzzy set retrieval).

Результати інформаційного пошуку повинні відповісти таким вимогам:

- **релевантність** (від англ. Relevant) – стосується результатів роботи пошукової системи і експертної системи; ступінь відповідності запиту і знайденого, тобто доречність результату. Одне з найбільш близьких понять «релевантності» – «адекватність», тобто оцінка ступеня відповідності, практичної та соціальної застосовності результату варіантів вирішення завдання.
- **пертінентність** (від англ. Pertinent) – співвідношення обсягу корисної інформації до загального обсягу отриманої інформації.

18. Пошукові індекси.

https://uk.wikipedia.org/wiki/Пошуковий_індекс

https://ru.wikipedia.org/wiki/Инвертированный_индекс

Пошуковий індекс — структура даних, яка містить інформацію про документи та використовується в пошукових системах. Індексування, що здійснюється **пошуковою машиною**, — процес збору, сортування та зберігання даних з метою забезпечення швидкого та точного **пошуку інформації**. Створення індексу включає міждисциплінарні поняття з лінгвістики, когнітивної психології, математики, інформатики та фізики. Веб-індексуванням називають процес індексування в контексті пошукових машин, розроблених для пошуку **веб-сторінок** в Інтернеті.

Індексні структури даних

Архітектура пошукової системи розрізняється за способами індексування і за методами зберігання індексів, задовольняючи чинники^[↗]. Індекси бувають наступних типів:

Суфіксне дерево

Образно структуроване як **дерево**, підтримує лінійний час пошуку. Побудовано на зберіганні суфіксів слів. Дерева підтримують розширене хешування, яке важливо для індексації пошукової системи^[8]. Використовується для пошуку за шаблоном в послідовностях **ДНК** та **клasterизації**. Основним недоліком є те, що зберігання слова в дереві може потребувати простір більший, ніж необхідно для зберігання самого слова^[9]. Альтернативний запис — **суфіксний масив**^[ru]. Вважається, що він вимагає менше віртуальної пам'яті та підтримує **блочно-сортувальний стиск** даних.

Інвертований індекс

Сховище списку входжень кожного критерію пошуку^[10], зазвичай у формі **хеш-таблиць** або **бінарного дерева**^{[11][12]}.

Індекс цитування

Сховище цитат або гіперпосилань між документами для підтримки аналізу цитування, предмет бібліометрії.

N-грами

Сховище послідовностей довжин, даних для підтримки інших типів пошуку або аналізу тексту^[13].

Матриця термів документа

Використовується в латентно-семантичному аналізі (ЛСА), зберігає входження слів у документах двовимірної розрідженої матриці.

Інвертований індекс (англ. *inverted index*) — структура даних, в якій для кожного слова колекції документів у відповідному списку перераховані всі документи в колекції — в яких воно зустрілося. Інвертований індекс використовується для пошуку за текстами.

Є два варіанти інвертованого індексу:

- індекс, який містить лише список документів для кожного слова,
- індекс, додатково включає позицію слова в кожному документу^[1].

Застосування [ред. | ред. код]

Опишемо, як вирішується завдання знаходження документів, в яких зустрічаються всі слова з пошукового запиту. При обробці однословінного пошукового запиту відповідь вже є в інвертованому індексі — достатньо взяти список, відповідний слову із запиту. При обробці багатословінної запиту беруться списки, відповідні кожному зі слів запиту і пересічні.

Зазвичай в пошукових системах після побудови за допомогою інвертованого індексу списку документів, що містять слова із запиту, йде ранжування документів зі списку. Інвертований індекс — це найпопулярніша структура даних, яка використовується в інформаційному пошуку^[2].

Приклад [ред. | ред. код]

Нехай у нас є корпус з трьох текстів $T_0 = \text{"it is what it is"}$, $T_1 = \text{"what is it"}$ та $T_2 = \text{"it is a banana"}$, тоді інвертований індекс буде виглядати наступним чином:

```
"a":      {2}
"banana": {2}
"is":     {0, 1, 2}
"it":     {0, 1, 2}
"what":   {0, 1}
```

Тут цифри позначають номери текстів, у яких зустрілося відповідне слово. Тоді відпрацювання пошукового "what is it" запиту дастъ наступний результат $\{0,1\} \cap \{0,1,2\} \cap \{0,1,2\} = \{0,1\}$.

19. Ранжоване отримання інформації.

https://uk.wikipedia.org/wiki/Навчання_ранжуванню

https://en.wikipedia.org/wiki/Learning_to_rank

[https://en.wikipedia.org/wiki/Ranking_\(information_retrieval\)](https://en.wikipedia.org/wiki/Ranking_(information_retrieval))

Рейтинг результатів запиту є однією з основних задач інформаційного пошуку (ІП), яка є науковою та інженерною дисципліною яка використовується пошуковими системами. Для заданого запиту q і колекції документів D , які відповідають запиту, задача полягає у присвоєнні рейтингу цим документам, тобто потрібно відсортувати, документи в D відповідно до певного критерію, так, щоб «найкращі» результати з'являться на початку списку результатів, який відображається користувачу. У класичному варіанті критерію рейтингу формулюються у термінах релевантності документів відповідно до інформаційної потреби, яка виражена у запиті.

Рейтинг часто зводиться до розрахунку числових балів для пар запит-документ. Для цього використовується базова функція ранжування — це може бути косинус подібності між векторами tf-idf, [1] які відповідають запиту і документу у векторній моделі, бали в BM25^[en] або ймовірності у ймовірнісній моделі інформаційного пошуку. Потім рейтинг може бути розрахований шляхом сортування документів у порядку зменшення балів. Інший підхід полягає у визначенні функції оцінки на парі документів d_1, d_2 , яка приймає позитивне значення тоді і тільки тоді, коли d_1 більш релевантний запиту, ніж d_2 , і використати цю інформацію для сортування.

Функції ранжування оцінюються різними методами. Один з найпростіших — це визначити точність перших k найвищих результатів для деяких фіксованих k . Наприклад, це може бути частка 10 найкращих результатів, які є релевантними, в середньому за багатьма запитами.

Булевы модели

Булева модель или BIR - это простая базовая модель запроса, в которой каждый запрос следует базовым принципам реляционной алгебры с алгебраическими выражениями и где документы

выбираются, если они не полностью совпадают друг с другом. Поскольку запрос либо извлекает документ (1), либо не извлекает документ (0), методологии для их ранжирования не существует.

Модель векторного пространства

Поскольку булева модель выбирает только полные совпадения, проблема не решается, если документы частично совпадают. Модель векторного пространства решает эту проблему путем введения векторов элементов индекса, каждому из которых присвоен вес. Веса варьируются от положительного (если сопоставлено полностью или в некоторой степени) до отрицательного (если не сопоставлено или полностью сопоставлено), если документы присутствуют. Частота терминов - обратная частота документов (tf-idf) является одним из самых популярных методов, где весовые коэффициенты - это термины (например, слова, ключевые слова, фразы и т. д.), А размерность - количество слов в корпусе.

Оценка сходства между запросом и документом может быть найдена путем вычисления значения косинуса между вектором веса запроса и вектором веса документа с использованием сходства косинуса. Желаемые документы могут быть выбраны путем ранжирования их в соответствии с показателем сходства и извлечением лучших k документов, которые имеют наивысшие оценки или наиболее релевантны для вектора запроса.

Вероятностная модель

В вероятностной модели ранжирование основано на вероятности, которая дает интерпретации, в которых запросы представлены в виде независимых логических векторов. Okapi BM25 предоставляет функцию ранжирования для ранжирования документов, имеющих отношение к запросу.

Evaluation Measures [edit]

The most common measures of evaluation are precision, recall, and f-score. They are computed using unordered sets of documents. These measures must be extended, or new measures must be defined, in order to evaluate the ranked retrieval results that are standard in modern search engines. In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top k retrieved documents. For each such set, precision and recall values can be plotted to give a precision-recall curve.^[5]

Precision [edit]

Precision measures the exactness of the retrieval process. If the actual set of relevant documents is denoted by I and the retrieved set of documents is denoted by O, then the precision is given by:

$$\text{Precision} = \frac{|\{I\} \cap \{O\}|}{|\{O\}|}$$

Recall [edit]

Recall is a measure of completeness of the IR process. If the actual set of relevant documents is denoted by I and the retrieved set of documents is denoted by O, then the recall is given by:

$$\text{Recall} = \frac{|\{I\} \cap \{O\}|}{|\{I\}|}$$

F1 Score [edit]

F1 Score tries to combine the precision and recall measure. It is the harmonic mean of the two. If P is the precision and R is the recall then the F-Score is given by:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Алгоритм PageRank выводит распределение вероятностей, используемое для представления вероятности того, что человек, случайно щелкнув по ссылкам, попадет на любую конкретную страницу. PageRank может быть рассчитан для коллекций документов любого размера. В нескольких исследовательских работах предполагается, что распределение равномерно распределяется между всеми документами в коллекции в начале вычислительного процесса. Расчеты PageRank требуют нескольких проходов через коллекцию, чтобы скорректировать приблизительные значения PageRank для более точного отражения теоретического истинного значения. Формулы приведены ниже:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

то есть значение PageRank для страницы u зависит от значений PageRank для каждой страницы v , содержащейся в наборе B_u (наборе, содержащем все страницы, ссылающиеся на страницу u), деленном на число $L(v)$ ссылок со страницы v .

20. Архітектура пошукової системи.

<https://web.njit.edu/~alexg/courses/cs345/OLD/F15/solutions/f2345f15.pdf>

https://uk.wikipedia.org/wiki/Пошукова_система

<https://www.slideshare.net/SylvainUtard/architecture-of-a-search-engine-paris-tech-talks-7>

Любая архитектура поисковой системы должна удовлетворять двум основным критериям.

- 1.1.1 Эффективность (Качество), которая будет удовлетворять критерию релевантности.
- 1.1.2 Эффективность (скорость), которая будет удовлетворять требованиям времени отклика и пропускной способности, т. Е. Обрабатывать как можно больше запросов максимально быстро. С этим связано понятие масштабируемости.

Как работает поисковая система



Высокоуровневая архитектура стандартного краулера

Основные составляющие поисковой системы: [поисковый робот](#), [индексатор](#), [поисковик](#)^[14].

Как правило, системы работают поэтапно. Сначала поисковый робот получает контент, затем индексатор генерирует доступный для поиска индекс, и наконец, поисковик обеспечивает функциональность для поиска индексируемых данных. Чтобы обновить поисковую систему, этот цикл индексации выполняется повторно^[14].

Поисковые системы работают, храня информацию о многих веб-страницах, которые они получают из [HTML](#)-страниц. Поисковый робот или «краулер» ([англ. Crawler](#)) — программа, которая автоматически проходит по всем ссылкам, найденным на странице, и выделяет их. Краулер, основываясь на ссылках или исходя из заранее заданного списка адресов, осуществляет поиск новых документов, ещё не известных поисковой системе. Владелец сайта может исключить определённые страницы при помощи [robots.txt](#), используя который можно запретить индексацию файлов, страниц или каталогов сайта.

Поисковая система анализирует содержание каждой страницы для дальнейшего индексирования. Слова могут быть извлечены из заголовков, текста страницы или специальных полей — [метатегов](#). Индексатор — это модуль, который анализирует страницу, предварительно разбив её на части, применяя собственные лексические и морфологические алгоритмы. Все элементы веб-страницы вычленяются и анализируются отдельно. Данные о веб-страницах хранятся в индексной базе данных для использования в последующих запросах. Индекс позволяет быстро находить информацию по запросу пользователя^[15].

Ряд поисковых систем, подобных Google, хранят исходную страницу целиком или её часть, так называемый [кэш](#), а также различную информацию о веб-странице. Другие системы, подобные системе AltaVista, хранят каждое слово каждой найденной страницы. Использование кэша помогает ускорить извлечение информации с уже посещённых страниц^[15]. Кэшированные страницы всегда содержат тот текст, который пользователь задал в поисковом запросе. Это может быть полезно в том случае, когда веб-страница обновилась, то есть уже не содержит текст запроса пользователя, а страница в кэше ещё старая^[15]. Эта ситуация связана с потерей ссылок ([англ. linkrot](#)^[en]) и дружественным по отношению к пользователю ([юзабилити](#)) подходом Google. Это предполагает выдачу из кэша коротких фрагментов текста, содержащих текст запроса. Действует [принцип наименьшего](#)

[удивления](#), пользователь обычно ожидает увидеть искомые слова в текстах полученных страниц ([User expectations](#)^[en]). Кроме того, что использование кэшированных страниц ускоряет поиск, страницы в кэше могут содержать такую информацию, которая уже нигде более не доступна.

Поисковик работает с выходными файлами, полученными от индексатора. Поисковик принимает пользовательские запросы, обрабатывает их при помощи индекса и возвращает результаты поиска^[14].

Когда пользователь вводит запрос в поисковую систему (обычно при помощи [ключевых слов](#)), система проверяет свой индекс и выдаёт список наиболее подходящих веб-страниц (отсортированный по какому-либо критерию), обычно с краткой аннотацией, содержащей заголовок документа и иногда части текста^[15]. Поисковый индекс строится по специальной методике на основе информации, извлечённой из веб-страниц^[11]. С 2007 года поисковик Google позволяет искать с учётом времени, создания искомых документов (вызов меню «Инструменты поиска» и указание временного диапазона).

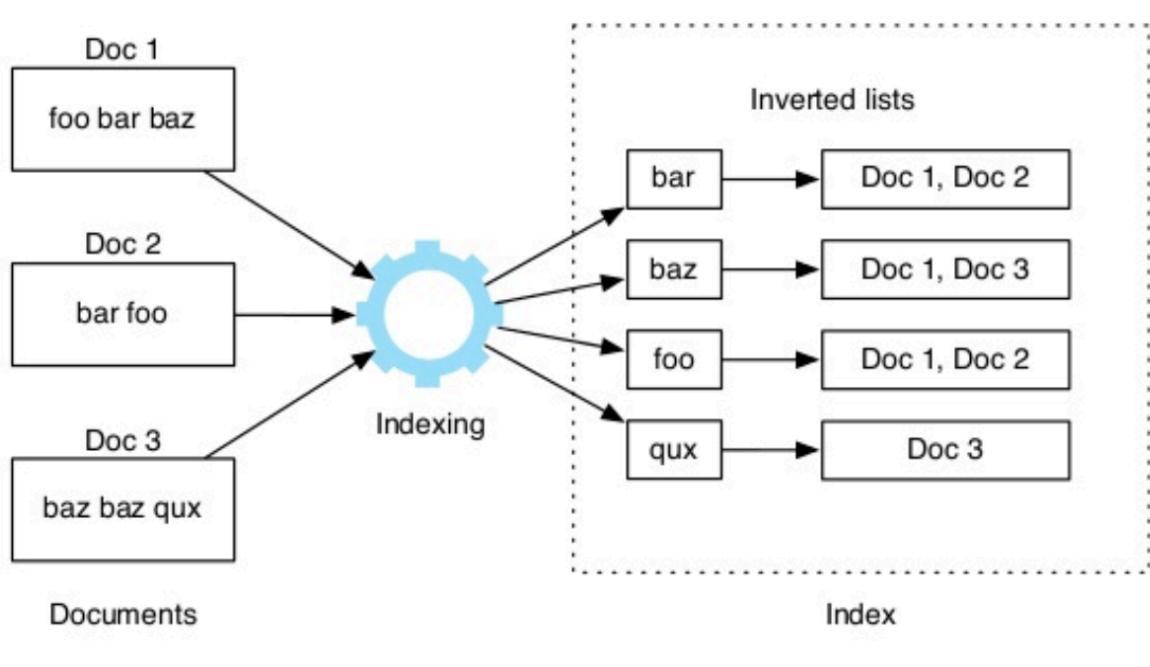
Большинство поисковых систем поддерживает использование в запросах [булевых операторов](#) И, ИЛИ, НЕ, что позволяет уточнить или расширить список искомых ключевых слов. При этом система будет искать слова или фразы точно так, как было введено. В некоторых поисковых системах есть возможность [приближённого поиска](#)^[en], в этом случае пользователи расширяют область поиска, указывая расстояние до ключевых слов^[15]. Есть также [концептуальный поиск](#)^[en], при котором используется статистический анализ употребления искомых слов и фраз в текстах веб-страниц. Эти системы позволяют составлять запросы на естественном языке. Примером такой поисковой системы является сайт [ask.com](#).

Полезность поисковой системы зависит от [релевантности](#) найденных ею страниц. Хоть миллионы веб-страниц и могут включать некое слово или фразу, но одни из них могут быть более релевантны, популярны или авторитетны, чем другие. Большинство поисковых систем использует методы [ранжирования](#), чтобы вывести в начало списка «лучшие» результаты. Поисковые системы решают, какие страницы более релевантны, и в каком порядке должны быть показаны результаты, по-разному^[15]. Методы поиска, как и сам Интернет со временем меняются. Так появились два основных типа поисковых

систем: системы предопределённых и иерархически упорядоченных ключевых слов и системы, в которых генерируется [инвертированный индекс](#) на основе анализа текста.

Большинство поисковых систем являются коммерческими предприятиями, которые получают прибыль за счёт [рекламы](#), в некоторых поисковиках можно купить за отдельную плату первые места в выдаче для заданных ключевых слов. Те поисковые системы, которые не берут денег за порядок выдачи результатов, зарабатывают на контекстной рекламе, при этом рекламные сообщения соответствуют запросу пользователя. Такая реклама выводится на странице со списком результатов поиска, и поисковики зарабатывают при каждом клике пользователя на рекламные сообщения.

Implementation: Indexing process



Типы поисковых систем

Существует четыре типа поисковых систем: с поисковыми роботами, управляемые человеком, гибридные и мета-системы^[16].

- *системы, использующие поисковые роботы*

Состоят из трёх частей: [краулер](#) («бот», «робот» или «паук»), [индекс](#) и программное обеспечение поисковой системы. Краулер нужен для обхода сети и создания списков веб-страниц. Индекс — большой архив

копий веб-страниц. Цель программного обеспечения — оценивать результаты поиска. Благодаря тому, что поисковый робот в этом механизме постоянно исследует сеть, информация в большей степени актуальна. Большинство современных поисковых систем являются системами данного типа.

- системы, управляемые человеком (каталоги ресурсов)

Эти поисковые системы получают списки веб-страниц. Каталог содержит адрес, заголовок и краткое описание сайта. Каталог ресурсов ищет результаты только из описаний страницы, представленных ему веб-мастерами. Достоинство каталогов в том, что все ресурсы проверяются вручную, следовательно, и качество контента будет лучше по сравнению с результатами, полученными системой первого типа автоматически. Но есть и недостаток — обновление данных каталогов выполняется вручную и может существенно отставать от реального положения дел. Ранжирование страниц не может мгновенно меняться. В качестве примеров таких систем можно привести [каталог Yahoo](#)^[en], [dmoz](#) и [Galaxy](#).

- гибридные системы

Такие поисковые системы, как [Yahoo](#), [Google](#), [MSN](#), сочетают в себе функции систем, использующие поисковых роботов, и систем, управляемых человеком.

- мета-системы

Метапоисковые системы объединяют и ранжируют результаты сразу нескольких поисковиков. Эти поисковые системы были полезны, когда у каждой поисковой системы был уникальный индекс, и поисковые системы были менее «умными». Поскольку сейчас поиск намного улучшился, потребность в них уменьшилась. Примеры: [MetaCrawler](#)^[en] и [MSN Search](#).

21. Методи прискорення пошуку текстів за запитом.

<https://www.statisticshowto.datasciencecentral.com/gibbs-sampling/>

https://en.wikipedia.org/wiki/Gibbs_sampling

<https://medium.com/@tomar.ankur287/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045>

Выборка Гиббса (также называемая чередующейся условной выборкой) представляет собой алгоритм Марковской цепочки Монте-Карло для многомерных данных, таких как обработка изображений и микрочипы. Он называется Монте-Карло, потому что он выбирает образцы из определенных распределений вероятности; цепь Маркова происходит из того факта, что каждый образец зависит от предыдущего образца. Выборка Гиббса относительно проста в реализации. Однако он менее эффективен, чем прямое моделирование из дистрибутива.

Метод основан на апостериорной плотности каждого параметра, в зависимости от других параметров. Это дает точное представление о предельных задних плотностях. Чтобы алгоритм работал, вы должны иметь возможность выбирать из всех условных распределений для всех параметров модели (Segura & Braun, 2004). На практике это не всегда возможно, поэтому необходимые условные распределения должны быть аппроксимированы другим алгоритмом, таким как Metropolis Hastings или выборка срезов.

Как это устроено

Выборка Гиббса - это эффективный способ сведения многомерной задачи к задаче меньшего размера. Весь вектор параметров делится на меньшие подвекторы (например, векторы с одним параметром). Одна итерация алгоритма приводит к тому, что каждый субвектор выбирается случайным образом с использованием апостериорной плотности субвектора, в зависимости от текущих значений другого субвектора (Duchateau & Janssen, 2007, p.234). В тех случаях, когда вектор параметров очень велик и подразделяется на очень мелкие фрагменты, для сходимости может потребоваться очень много времени.

$$p(z_{d,n} = k | \vec{z}_{-d,n}, \vec{w}, \alpha, \lambda) = \frac{n_{d,k} + \alpha_k}{\sum_i^K n_{d,i} + \alpha_i} \frac{v_{k,w_{d,n}} + \lambda_{w_{d,n}}}{\sum_i v_{k,i} + \lambda_i}$$

Credit: [Jordan Boyd-Graber](#)

где:

$n(d, k)$: сколько раз документ d использовал тему k

$v(k, w)$: количество раз, когда тема k использует данное слово

α_k : параметр Дирихле для распространения документа по теме

λ_w : параметр Дирихле для рассылки по темам

22. Машинний переклад.

https://uk.wikipedia.org/wiki/Машинний_переклад

Машинний переклад (МП) — технології автоматизованого перекладу текстів (письмових та усних) з однієї природної мови на іншу за допомогою комп'ютера; напрямок наукових досліджень, пов'язаний з побудовою систем автоматизованого перекладу.

На базовому рівні, робота комп'ютерних програм для перекладу полягає у заміні слів чи словосполучень з однієї мови на слова чи словосполучення з іншої. Однак тоді виникає проблема, що така заміна не може забезпечити якісний переклад тексту, адже потрібне визначення та розпізнання слів та цілих фраз з мови оригіналу. Це спонукає активну наукову діяльність у галузі комп'ютерної лінгвістики. Наразі, для вирішення неоднозначностей при перекладі, використовуються багатомовні онтологічні ресурси, такі як WordNet та UWN.

Машинний переклад - одна з підгруп комп'ютерної лінгвістики, яка досліджує використання програмного забезпечення для перекладу тексту з однієї мови на іншу. На початковому рівні МТ виконує звичайну заміну слів з однієї мови на слова з іншої мови, але, зазвичай, переклад здійснений таким чином не є дуже якісним, адже для того щоб, повністю передати сенс речення, та знайти найспорідненніший аналог в «цільовій» (target language) - потрібній перекладачу мові, часто потрібно здійснювати переклад цілої фрази.

Існують два принципово різних підходи до побудови алгоритмів машинного перекладу: заснований на правилах (rule-based) і статистичний, або заснований на статистиці (statistical-based).

МП на основі правил (Rule-based MT)

МП на основі правил ((Rule-based MT) - [RBMT], «Класичний підхід» МП) - система машинного перекладу сформована на базі лінгвістичної інформації з одномовних (*unilingual*), двомовних (*bilingual*) чи

багатомовних (*multilingual*) словників та граматичних правил вихідної мови та цільової мови.

Система охоплює основні семантичні, морфологічні та синтаксичні закономірності кожної мови. Відповідно, для того щоб здійснити переклад, система повинна зробити попередній морфологічний, синтаксичний та семантичний аналіз тексту, і тільки після цього вона генерує речення. Найбільший мінус RB-перекладу полягає в тому, що для здійснення програмою коректного перекладу, її база даних повинна містити усі орфографічні варіації та помилкові форми введення слів, а для всіх випадків неоднозначності повинні бути написані правила лексичного відбору.

Сама по собі, адаптація до нових доменів є не таким вже і складним процесом, оскільки основи граматики для всіх доменів однакові, а налаштування сфер користувачкої діяльності обмежується лише корекцією лексичного відбору.

Отож, така система машинного перекладу є першим, класичним методом його здійснення, вона дозволяє отримати більш якісний результат, аніж статистичний метод але синтезує переклад повільніше.

Статистичний переклад (Statistical MT)

Статистичний машинний переклад — це різновид машинного перекладу тексту, заснований на порівнянні великих обсягів мовних пар. Мовні пари — тексти, що містять речення однією мовою і відповідні речення іншою, можуть бути як варіантами написання двох речень людиною — носієм двох мов, так і набором речень та їх перекладів, виконаних людиною. Таким чином статистичний машинний переклад володіє властивістю «самонавчання». Чим більше в розпорядженні програми є мовних пар і чим точніше вони відповідають один одному, тим кращий результат статистичного машинного перекладу.

Під поняттям «статистичного машинного перекладу» мається на увазі загальний підхід до вирішення проблеми перекладу, який заснований на пошуку найімовірнішого перекладу речення з використанням даних, отриманих з двомовної сукупності текстів. Як приклад двомовної сукупності текстів можна назвати парламентські звіти, які являють

собою протоколи дебатів в парламенті. Двомовні парламентські звіти видаються в Канаді, Гонконгу та інших країнах; офіційні документи Європейського економічного співтовариства видаються 11 мовами; а Організація Об'єднаних Націй публікує документи на декількох мовах. Як виявилося, ці матеріали є безцінними ресурсами для статистичного машинного перекладу.

Дана система базується на статистичному вирахуванні вірогідності збігів. Задля виконання перекладу програма повинна мати доступ до сотень мільйонів документів, які заздалегідь були перекладені людьми. Такі документи слугують для системи шаблонами, на основі яких вона і здійснює переклад. Чим більше документів, тим вища ймовірність більш якісного перекладу.

На початку свого існування, з 2006 року, Google Translate базувався саме на статистичному методі машинного перекладу, і здійснений ним переклад був дуже низької якості, і вважався одним з найгірших варіантів перекладу, який може здійснити онлайн-перекладач. Сьогодні Google використовую «нейронний» метод МП і складає серйозну конкуренцію комерційним підприємствам, продукція яких не є безкоштовною.

Гіbridний МП (Hybrid MT)

Останніми роками все більшої популярності набирає Гіbridний МП (Hybrid machine translation [HMT]).

ГМП - використовує сильні сторони обох систем машинного перекладу, в результаті користувач отримує якісний переклад, який забезпечує RBMT та високу швидкість, яку надає статистичний метод.

23. Системи підтримки діалогу.

https://translate.google.com/translate?hl=uk&sl=en&u=https://en.wikipedia.org/wiki/Dialogue_system

Система діалогу або **розмовний агент** (**СА**) - це комп'ютерна система, призначена для спілкування з людиною з когерентною структурою. Системи діалогу використовують текст, мовлення, графіку, звуки, жести та інші режими для зв'язку як на вхідному, так і на вихідному каналі.

Contents

Після діалогових систем, що базувалися тільки на обробці письмового тексту, починаючи з початку шістдесятих років [1] , перша розмовна система діалогу була випущена проектом **DARPA** в США в 1977 році [2] . Після завершення цього п'ятирічного проекту деякі європейські проекти видавали першу систему діалогу, здатну говорити багатьма мовами (також французькою, німецькою та італійською) [3] . Ці перші системи використовувалися в індустрії телекомунікацій для надання різноманітних послуг телефону в певних областях, наприклад, автоматизований порядок денний та обслуговування таблиць поїздів.

Компоненти

Існує багато різних архітектур для систем діалогу. Які набори компонентів включені в систему діалогу і як ці компоненти поділяють обов'язки відрізняються від системи до системи. Принциповою для будь-якої системи діалогу є **менеджер діалогу** , який є компонентом, який керує станом діалогу і діалоговою стратегією. Типовий цикл діяльності в системі діалогу містить наступні етапи: [4]

1. Користувач розмовляє, і вхід перетворюється на звичайний текст за допомогою **розвізнавача / декодера входу** системи, який може включати:
 - [автоматичний розпізнавач мовлення](#) (ASR)
 - [розвізнавач жестів](#)

- [роздільник рукописного тексту](#)
2. Текст аналізується підрозділом для **розуміння природної мови** (NLU), який може включати:
- Ідентифікація належного імені
 - [частини міток мовлення](#)
 - Синтаксичний / семантичний [парсер](#)
3. Семантичну інформацію аналізує [менеджер діалогу](#), який зберігає історію та стан діалогу і управляє загальним потоком розмови.
4. Зазвичай, менеджер діалогів контактує з одним або кількома **менеджерами завдань**, які мають знання конкретного домену завдання.
5. Менеджер діалогів виробляє висновок за допомогою **генератора вихідних даних**, який може включати:
- [генератор природної мови](#)
 - [генератор жестів](#)
 - [менеджер компонування](#)
6. Нарешті, висновок виводиться за допомогою **візуалізації виводу**, яка може включати:
- механізм перетворення [тексту в мову](#) (TTS)
 - [Говорити голову](#)
 - [робота](#) або [аватара](#)

Системи діалогу, засновані на текстовому інтерфейсі (наприклад, текстовий чат), містять лише етапи 2–5.

Типи систем

Системи діалогу поділяються на такі категорії, які перераховані тут за кількома вимірами. Багато з категорій перекриваються, і відмінності можуть бути недостатньо встановленими.

- за [модальністю](#)
 - [на основі тексту](#)
 - [система розмовного діалогу](#)

- [графічний інтерфейс користувача](#)
 - [мультимодальний](#)
- за допомогою пристрою
 - телефонні системи
 - Системи [КПК](#)
 - системи автомобіля
 - [робототехнічні](#) системи
 - [настільні / портативні](#) системи
 - рідною
 - в [браузерних](#) системах
 - у [віртуальній машині](#)
 - у [віртуальному середовищі](#)
 - роботів
- за стилем
 - на основі команд
 - меню -переваги
 - [природна мова](#)
 - мовні графіті
- за ініціативою
 - ініціативи системи
 - ініціативу користувача
 - змішана ініціатива

24. Системи типу «Питання-Відповідь».

<https://ru.wikipedia.org/wiki/>

%D0%92%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D0%BD

%D0%BE-

%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%BD%D0%BA%

%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC

%D0%B0

<https://medium.com/lingvo-masino/question-and-answering-in-natural-language-processing-part-i-168f00291856>

<https://towardsdatascience.com/nlp-building-a-question-answering-model-ed0529a68c54>

<https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507>

Вопросно-ответная система (**QA-система**; от англ. *QA* – англ. *Question-answering system*) – **информационная система**, способная принимать вопросы и отвечать на них на **естественном языке**, другими словами, это система с естественно-языковым интерфейсом.

Классификация

Вопросно-ответные системы можно условно разделить на:

- **Узкоспециализированные** QA-системы работают в конкретных областях (например, медицина или обслуживание автомобилей).
 - **Общие** QA-системы работают с информацией по всем областям знаний, таким образом появляется возможность вести поиск в смежных областях.

• Архитектура

Первые QA-системы^[1] были разработаны в 1960-х годах и являлись естественно-языковыми оболочками для экспертных систем, ориентированных на конкретные области. Современные системы предназначаются для поиска ответов на вопросы в предоставляемых документах с использованием технологий обработки естественных языков (NLP).

Современные QA-системы обычно включают особый модуль — **классификатор вопросов**, который определяет тип вопроса и, соответственно, ожидаемого ответа. После этого анализа система постепенно применяет к предоставленным документам все более сложные и тонкие методы NLP, отбрасывая ненужную информацию. Самый грубый метод — **поиск в документах** — предполагает использование системы поиска информации для отбора частей текста, потенциально содержащих ответ. Затем **фильтр** выделяет фразы, похожие на ожидаемый ответ (например, на вопрос «Кто ...» фильтр вернет кусочки текста, содержащие имена людей). И, наконец, модуль **выделения ответов** найдет среди этих фраз правильный ответ.

Открытые наборы данных доступны для ответов на вопросы

- Стэнфордский набор данных для ответов на вопросы (SQuAD) [2] - это набор данных для понимания прочитанного, состоящий из вопросов, задаваемых толпочниками в ряде статей Википедии, где ответ на каждый вопрос представляет собой фрагмент текста или промежуток из соответствующего отрывка для чтения, или вопрос может быть без ответа. Здесь есть отличная статья.
- Набор данных WikiQA [3] - это общедоступный набор пар вопросов и ответов, собранных и аннотированных для исследования ответов на открытые вопросы. Он построен с использованием более естественного процесса и более чем на порядок больше, чем предыдущий набор данных. Кроме того, набор данных WikiQA также включает вопросы, для которых нет правильных предложений, что позволяет исследователям работать над инициированием ответов, критическим компонентом в любой системе обеспечения качества.
- Набор данных TREC-QA содержит шаблоны вопросов и ответов, а также пул документов, возвращаемых участвующими группами.

- Набор данных NewsQA [4] призван помочь исследовательскому сообществу в создании алгоритмов, способных отвечать на вопросы, требующие понимания на уровне человека и умения рассуждать. Используя статьи CNN из набора данных DeepMind Q & A, авторы подготовили краудсорсинговый набор данных для машинного чтения из 120K пар вопросов и ответов.

Типы вопросов и ответов

Существует три основных современных парадигмы ответа на вопрос:

- а) Факторный вопрос на основе IR Цель ответа на вопрос состоит в том, чтобы ответить на вопрос пользователя путем поиска коротких текстовых сегментов в Интернете или некоторой другой коллекции документов. На этапе обработки вопросов извлекается ряд информации из вопроса. Тип ответа указывает тип объекта, из которого состоит ответ (человек, местоположение, время и т. д.). В запросе указываются ключевые слова, которые должны использоваться системой IR для поиска документов.
- б) Ответ на вопрос, основанный на знаниях, - это идея ответа на вопрос на естественном языке путем сопоставления его с запросом по структурированной базе данных. Логическая форма вопроса, таким образом, либо в форме запроса, либо может быть легко преобразована в один. База данных может быть полной реляционной базой данных или более простыми структурированными базами данных, такими как наборы троек RDF. Системы для преобразования текстовой строки в любую логическую форму называются семантическими анализаторами. Семантические парсеры для ответов на вопросы обычно отображаются либо на какую-то версию исчисления предикатов, либо на язык запросов, такой как SQL или SPARQL.
- с) Использование нескольких источников информации: система IBM Watson [5,6] от IBM, выигравшая Jeopardy! Задача в 2011 году является примером системы, которая опирается на широкий спектр ресурсов для ответа на вопросы. Первый этап - обработка вопросов. Система DeepQA выполняет анализ, маркировку именованных сущностей и извлечение отношений по данному вопросу. Затем, подобно текстовым

системам, система DeepQA извлекает фокус, тип ответа (также называемый лексическим типом ответа или LAT) и выполняет классификацию вопросов и разделение вопросов. Далее DeepQA извлекает фокус вопроса. Наконец, вопрос классифицируется по типу как вопрос определения, множественный выбор, головоломка или заполнение пробела. Далее следует этап генерации ответов кандидата в соответствии с типом вопроса, где обработанный вопрос объединяется с внешними документами и другими источниками знаний, чтобы предложить множество ответов кандидатов. Эти кандидатские ответы могут быть извлечены из текстовых документов или из структурированных баз знаний. Затем он проходит через этап подсчета ответов кандидата, который использует множество источников доказательств для оценки кандидатов. Одним из наиболее важных является лексический тип ответа. На последнем этапе объединения и подсчета ответов сначала объединяются ответы кандидатов, которые эквивалентны. Слияние и ранжирование фактически выполняется итеративно; сначала кандидаты ранжируются классификатором, давая приблизительное первое значение для каждого ответа кандидата, затем это значение используется, чтобы решить, какой из вариантов имени выбрать в качестве объединенного ответа, затем объединенные ответы переупорядочиваются.

25. Розпізнавання іменованих сутностей текстів за допомогою Умовних випадкових полів.

<http://dspace.nbuv.gov.ua/bitstream/handle/123456789/126400/16-Marchenko.pdf?sequence=1>

Класифікатор на основі моделі умовних випадкових полів – Conditional random fields

Метод умовних випадкових полів – Conditional random field (CRF) є аналогом методу марковських випадкових полів (Markov random fields). Даний метод користується широкою популярністю у різних областях штучного інтелекту. Зокрема його успішно використовують у задачах розпізнавання мовлення та образів, в обробці текстової інформації, у комп’ютерній графіці та в інших задачах.

Марковським випадковим полем називають графову модель, яка використовується для представлення сумісних розподілів набору декількох випадкових змінних. Формально марковське випадкове поле складається з наступних компонентів:

- неорієнтований граф або фактор-граф $G = (V, E)$, де кожна вершина $v \in V$ – випадкова змінна X і кожне ребро $(u, v) \in E$ – залежність між випадковими величинами u і v ;
- набір потенційних функцій (potential function) або факторів $\{\varphi_k\}$, одна для кожної кліки у графі (кліка – повний підграф G неорієнтованого графу). Функція φ_k ставить кожному можливому стану елементів кліки у відповідність деяке невід’ємне дійсне число.

Вершини, що не є суміжними, мають відповідати умовно незалежним випадковим величинам. Група суміжних вершин формує кліку, набір станів вершин є аргументом відповідної потенційної функції.

Сумісний розподіл набору випадкових величин $X = \{x_k\}$ у марковському випадковому полі обчислюється за формулою:

$$P(x) = \frac{1}{Z} \prod_k \varphi_k(x_{\{k\}}),$$

де $\varphi_k(x_{\{k\}})$ – потенційна функція, що описує стан випадкових величин у k -ій кліці; Z – коефіцієнт нормалізації, що обчислюється за формулою:

$$Z = \sum_{x \in X} \prod_k \varphi_k(x_{\{k\}}).$$

Множина вхідних лексем $X = \{x_t\}$ та множина відповідних їм типів $Y = \{y_t\}$ у сукупності формують множину випадкових змінних $V = X \cup Y$. Для розв'язання задачі виділення інформації з тексту достатньо визначити умовну ймовірність $P(Y | X)$. Потенційна функція має вигляд:

$$\varphi_k(x_{\{k\}}) = \exp\left(\sum_k \lambda_k f_k(y_t, y_{t-1}, x_t)\right),$$

де $\sum \{\lambda_k\}$ – дійснозначний параметричний вектор (множники Лагранжа), $\sum \{f_k(y_t, y_{t-1}, x_t)\}$ – набір ознакових функцій. Тоді лінійним умовним випадковим полем називається розподіл виду:

$$p(y | x) = \frac{1}{Z(x)} \prod_k \exp\left(\sum_k \lambda_k f_k(y_t, y_{t-1}, x_t)\right).$$

Коефіцієнт нормалізації $Z(x)$ обчислюється за формулою:

$$Z(x) = \sum_{y \in Y} \prod_k \exp\left(\sum_k \lambda_k f_k(y_t, y_{t-1}, x_t)\right).$$

Обчислення моделі $p(y | x)$ відбувається як розв'язання оптимізаційної задачі з заданими обмеженнями [2] (різниця між спостереженням та його оцінкою має бути нульовою та має виконуватися умова

$$\sum_{y \in Y} p(y | x) = 1 \text{ по всім } x \in X.$$

На кожній ітерації заново обчислюються множники Лагранжа, обчислення проводиться з використанням традиційних алгоритмів – «forward-backward» та Вітербі.

Метод CRF, як і метод марковські моделі максимальної ентропії (MMME), є дискримінтивним імовірнісним методом, на відміну від генеративних методів, таких як приховані марковські моделі НММ та модель Байеса (Naïve Bayes).

За аналогією з марковськими моделями максимальної ентропії, вибір факторів-ознак для завдання імовірності переходу між станами при наявності спостереження значення x_t залежить від специфіки конкретних даних, але на відміну від того ж MMME, CRF може враховувати будь-які особливості та взаємозв'язки у вхідних даних. Вектор ознак $\Lambda = \{\lambda_k\}$ обчислюється на основі навчальної вибірки та визначає вагу кожної потенційної функції.

В умовних випадкових полях відсутня так звана *label bias problem* – ситуація, коли перевагу мають стани з меншою кількістю переходів, так як буде один єдиний розподіл імовірностей та нормалізація (коефіцієнт $Z(x)$) виконується загалом, а не у рамках окремого стану. Це, безумовно, є перевагою метода: алгоритм не потребує припущення незалежності спостережних змінних. Крім того, використання довільних факторів дозволяє описати різноманітні ознаки об'єктів, що знижує вимоги до повноти та обсягу навчальної вибірки. При цьому точність буде визначатися не лише обсягом вибірки, але й обраними факторами.

Недоліком підходу CRF є обчислювальна складність аналізу навчальної вибірки, що ускладнює постійне оновлення моделі при отриманні нових навчальних даних. Слід відзначити високу швидкість роботи алгоритму CRF, що є дуже важливою перевагою при обробці великих обсягів інформації.

26. Методи пониження розмірності латентних семантичних моделей.

<https://www.statisticshowto.datasciencecentral.com/gibbs-sampling/>

https://en.wikipedia.org/wiki/Gibbs_sampling

<https://medium.com/@tomar.ankur287/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045>

Выборка Гиббса (также называемая чередующейся условной выборкой) представляет собой алгоритм Марковской цепочки Монте-Карло для многомерных данных, таких как обработка изображений и микрочипы. Он называется Монте-Карло, потому что он выбирает образцы из определенных распределений вероятности; цепь Маркова происходит из того факта, что каждый образец зависит от предыдущего образца. Выборка Гиббса относительно проста в реализации. Однако он менее эффективен, чем прямое моделирование из дистрибутива.

Метод основан на апостериорной плотности каждого параметра, в зависимости от других параметров. Это дает точное представление о предельных задних плотностях. Чтобы алгоритм работал, вы должны иметь возможность выбирать из всех условных распределений для всех параметров модели (Segura & Braun, 2004). На практике это не всегда возможно, поэтому необходимые условные распределения должны быть аппроксимированы другим алгоритмом, таким как Metropolis Hastings или выборка срезов.

Как это устроено

Выборка Гиббса - это эффективный способ сведения многомерной задачи к задаче меньшего размера. Весь вектор параметров делится на меньшие подвекторы (например, векторы с одним параметром). Одна итерация алгоритма приводит к тому, что каждый субвектор выбирается случайным образом с использованием апостериорной плотности субвектора, в зависимости от текущих значений другого субвектора (Duchateau & Janssen, 2007, p.234). В тех случаях, когда вектор параметров очень велик и подразделяется на очень мелкие фрагменты, для сходимости может потребоваться очень много времени.

$$p(z_{d,n} = k | \vec{z}_{-d,n}, \vec{w}, \alpha, \lambda) = \frac{n_{d,k} + \alpha_k}{\sum_i^K n_{d,i} + \alpha_i} \frac{v_{k,w_{d,n}} + \lambda_{w_{d,n}}}{\sum_i v_{k,i} + \lambda_i}$$

Credit: [Jordan Boyd-Graber](#)

где:

$n(d, k)$: сколько раз документ d использовал тему k

$v(k, w)$: количество раз, когда тема k использует данное слово

α_k : параметр Дирихле для распространения документа по теме

λ_w : параметр Дирихле для рассылки по темам

27. Методи автоматичної побудови таксономії.

https://en.wikipedia.org/wiki/Automatic_taxonomy_construction

Автоматична побудова таксономії (АТС) - це використання програм для генерації таксономічних класифікацій з сукупності текстів, які називаються корпусом. АТС - це гілка обробки природної мови, яка, в свою чергу, є гілкою штучного інтелекту.

Розробка і підтримка таксономії вручну - це трудомістке завдання, що вимагає значного часу і ресурсів, включаючи знайомство або досвід в області таксономії (предмет, область або набір елементів, які представляє таксономія). Крім того, розробники моделей доменів мають свої власні точки зору, які неминуче, навіть ненавмисно, пробиваються в таксономії. АТС використовує методи штучного інтелекту для автоматичного створення таксономії для домену, щоб уникнути цих проблем.

Taxonomies

Крім іншого, таксономія може використовуватися для організації та індексації знань (що зберігаються у вигляді документів, статей, відео і т. Д), Наприклад, у формі системи класифікації бібліотек, щоб користувачі могли знаходити інформацію, яку вони шукають. Таксономії, як правило, мають деревоподібну структуру і ділять домен (суб'єкт, поле або набір елементів, які представляє таксономія) на категорії, засновані на значенні властивостей, званих таксони [необхідне пояснення] [потрібно цитування]. Вперше таксономії використовувалися біологами [необхідна цитата] для угруповання організмів в біологічні класифікації, впорядковані за індивідуальними ранжир таксонам, специфічним для біології, таким як царство, тип, рід і види.

Гіпонімія і відносини "is-a"

У програмах УВС однією з найважливіших завдань є виявлення гіперних і гіпонімних зв'язків між словами. Один із способів зробити це за текстом - це пошук певних фраз, таких як «is a» і «наприклад».

У лінгвістиці відносини називаються фразеологізми. Слова, які описують категорії, називаються гиперонимов, а слова, які є прикладами категорій, є гіпонімії. Наприклад, собака є гіперним, а Фідо - одним з її гіпонімії. Слово може бути як гіпонімії, так і гіперним. Отже, собака є гіпонімії ссавця, а також гіперним Фідо.

Таксономії часто представляються у вигляді ієрархій «як є», де кожен рівень є більш конкретним (на математичній мові «підмножиною») рівень над ним. Наприклад, базова таксономія біології повинна мати такі поняття, як ссавець, яке є підмножиною тварин, і собаки і кішки, які є підмножинами ссавців. Цей вид таксономії називається моделлю, тому що конкретні об'єкти вважаються екземплярами концепції. Наприклад, Фідо - це приклад концептуальної собаки, а Пухнастий - це кішка. [1]

підходи

Є кілька підходів до УВС. Один з підходів полягає в тому, щоб використовувати правила для виявлення закономірностей в корпусі і використовувати ці закономірності для визначення таких відносин, як гіпонімія. Інші підходи використовують методи машинного навчання, такі як байесовский висновок і штучні нейронні мережі.

Системи УВС є ключовим компонентом навчання онтології і використовуються для автоматичного створення великих онтологій для таких областей, як страхування і фінанси. Вони також використовуються для поліпшення існуючих великих мереж, таких як Wordnet, щоб зробити їх більш повними і узгодженими.

28. Формальний концептуальний аналіз.

Аналіз формальних понять (АФП) (англ. Formal Concept Analysis, FCA) - гілка прикладної теорії алгебри решіток. Традиційно АФП відносять до області концептуальних структур в штучному інтелекті.

Аналіз формальних понять є методом аналізу даних. За допомогою цього методу аналізу можуть бути візуалізовані об'єктно-прізнакові залежності. Це досягається побудовою діаграми решітки формальних понять. Основна математична ідея аналізу формальних понять - можливість побудови повної решітки з будь-якого бінарним відношенням і формалізація опису поняття у вигляді пари (обсяг, зміст).

В основі решіток формальних понять лежить так зване відповідність Галуа, що задається на безлічі об'єктів і ознак і володіє відомим з філософського визначення понять властивістю зменшення обсягу зі зростанням змісту.

Основні визначення

Контекстом в АФП називають трійку $K = (G, M, I)$, де G - безліч об'єктів, M - безліч ознак, а ставлення $I \subseteq G \times M$ говорить про те, які об'єкти якими ознаками мають. Для довільних $A \subseteq G$ і $B \subseteq M$ визначені оператори Галуа:

$$A^+ = \{m \in M \mid \forall g \in A (g \mid m)\},$$

$$B^+ = \{g \in G \mid \forall m \in B (g \mid m)\}.$$

Оператор "(дворазове застосування оператора ') є оператором замикання: він ідемпотентів ($A'' = A''$), монотонний ($A \subseteq B$ тягне $A'' \subseteq B''$) і екстенсівен ($A \subseteq A''$). Безліч об'єктів $A \subseteq G$, таке, що $A'' = A$, називається замкнутим. Аналогічно для замкнутих множин ознак - підмножин безлічі M . Пара множин (A, B) , таких, що $A \subseteq G$, $B \subseteq M$, $A' = B$ і $B' = A$, називається формальним поняттям контексту K . Безліч A і B замкнуті і називаються обсягом і змістом формального поняття (A, B) відповідно. Для безлічі об'єктів A безліч іхніх спільних ознак A' служить описом подібності об'єктів з безлічі A , а замкнутий безліч A'' є кластером подібних об'єктів (з безліччю загальних ознак A'). Ставлення "бути більш загальним поняттям" задається наступним чином: $(A, B) \geq (C, D)$ тоді і тільки тоді, коли $A \supseteq C$.

Поняття формального контексту $K = (G, M, I)$, впорядковані по вкладенню обсягів утворюють решітку $B(G, M, I)$, звану гратами понять. Для візуалізації решіток понять використовують так звані діаграми Хассе, тобто граф покриття відносини "бути більш загальним поняттям".

29. Лінгвістичні моделі.

Лінгвістична модель — модель, яка на основі певних найзагальніших рис конкретної мови, формує деякі гіпотези про будову мови як абстрактну семіотичну систему, і встановлює відношення між результатами цих гіпотез та фактами реальних мов, що описуються конкретними лінгвістичними дисциплінами. Визначається властива даній мові парадигматична схема елементів (складових); компактне символічне зображення цієї схеми (або будь-яких частин чи елементів), і на основі цього створюється модель мови, як один із методів її вивчення (наприклад, породжуюча).

В залежності від наявних знань про мову і мети її вивчення, лінгвістичні моделі діляться на такі типи:

- дослідницькі — на основі тексту (і множини правильних фраз) моделюється діяльність лінгвіста.
- аналітичні — на основі граматики і словника моделюється розуміння тексту.
- синтетичні — на основі граматики і словника моделюється створення тексту.
- породжуючі — на основі граматики і словника моделюється вміння відрізняти правильне від неправильного в мові.

У природних мовах, як правило, невелика кількість правил визначає основну множину фактів, але для невеликої кількості фактів, що залишилися, які у більшості непродуктивні, потрібна дуже велика кількість правил. Тому інколи вигідніше використати не детерміністську (структурну) модель, яка може бути надмірно громіздкою через велику кількість правил, а ймовірнісну (статистичну) модель, яка потребує менше суто статистичних правил і тому вона є менш громіздкою. Можлива втрата точності правил компенсується в такій моделі її відносною простотою.

30. Генеративні лінгвістичні автомати.

Класифікація автоматов

Все автоматы, могут пониматься как машины Тьюринга, классифицированные по количеству, длине и движению лент, а также по используемым операциям чтения и записи. Термин «автомат дискретного состояния» иногда используется, чтобы подчеркнуть дискретную природу внутренних состояний. Основными классами являются преобразователи и акцепторы. В теории автоматов преобразователь - это автомат с входом и выходом; В качестве примера может служить любая машина Тьюринга для вычисления частично рекурсивной функции, как описано выше. Акцептор - это автомат без вывода, который в особом смысле распознает или принимает слова в машинном алфавите. Ввод акцептора записывается на ленту обычным способом, но в конце вычисления лента остается пустой, и принятие входного слова представляется специальным состоянием, называемым конечным состоянием. Таким образом, слово x или последовательность символов из алфавита, обозначенного буквой S , считается принятым акцептором A , если A вычисляет, начиная с начального состояния q_0 с x на ленте, и останавливается в конечном состоянии с лента полностью пуста. Подмножество, обозначенное U из набора слов S^* в алфавите S , называется принятым множеством, если существует автомат A , который принимает любое слово $x \in U$.

акцепторы

Элементарный результат теории автоматов состоит в том, что каждое рекурсивно перечислимое множество или диапазон частично рекурсивной функции является принятым множеством. В целом, акцепторы - это двусторонние неограниченные ленточные автоматы.

Полезная классификация акцепторов была введена в сочетании с теорией порождающих грамматик, разработанной в США лингвистом Ноамом Хомским. Генеративная грамматика - это система анализа, обычно идентифицируемая с лингвистикой. Таким образом, язык можно рассматривать как набор конечных правил, которые могут создавать предложения. Использование порождающей грамматики в контексте лингвистики или теории автоматов заключается в создании и демаркации всей грамматической конструкции языка, естественного или ориентированного на автоматы. Простая грамматика для фрагмента английского языка, определяемая 12 правилами (см. 7), может служить для представления основных идей.

(7)	<table border="0"> <tr> <td>1st rule:</td><td>$\overline{S} \rightarrow \overline{Pr} \ \overline{VP}$</td></tr> <tr> <td>2nd rule:</td><td>$\overline{NP} \rightarrow \overline{Art} \ \overline{Adj} \ \overline{N}$</td></tr> <tr> <td>3rd rule:</td><td>$\overline{Adj} \rightarrow \overline{Adv} \ \overline{Adj}$</td></tr> <tr> <td>4th rule:</td><td>$\overline{VP} \rightarrow \overline{V} \ \overline{NP}$</td></tr> <tr> <td>5th rule:</td><td>$\overline{Pr} \rightarrow \text{she}$</td></tr> <tr> <td>6th rule:</td><td>$\overline{Art} \rightarrow \text{a}$</td></tr> <tr> <td>7th rule:</td><td>$\overline{N} \rightarrow \overline{Adj} \ \overline{N}$</td></tr> <tr> <td>8th rule:</td><td>$\overline{Adj} \rightarrow \text{pretty}$</td></tr> <tr> <td>9th rule:</td><td>$\overline{Adj} \rightarrow \text{little}$</td></tr> <tr> <td>10th rule:</td><td>$\overline{Adv} \rightarrow \text{pretty}$</td></tr> <tr> <td>11th rule:</td><td>$\overline{N} \rightarrow \text{girl}$</td></tr> <tr> <td>12th rule:</td><td>$\overline{V} \rightarrow \text{is}$</td></tr> </table>	1st rule:	$\overline{S} \rightarrow \overline{Pr} \ \overline{VP}$	2nd rule:	$\overline{NP} \rightarrow \overline{Art} \ \overline{Adj} \ \overline{N}$	3rd rule:	$\overline{Adj} \rightarrow \overline{Adv} \ \overline{Adj}$	4th rule:	$\overline{VP} \rightarrow \overline{V} \ \overline{NP}$	5th rule:	$\overline{Pr} \rightarrow \text{she}$	6th rule:	$\overline{Art} \rightarrow \text{a}$	7th rule:	$\overline{N} \rightarrow \overline{Adj} \ \overline{N}$	8th rule:	$\overline{Adj} \rightarrow \text{pretty}$	9th rule:	$\overline{Adj} \rightarrow \text{little}$	10th rule:	$\overline{Adv} \rightarrow \text{pretty}$	11th rule:	$\overline{N} \rightarrow \text{girl}$	12th rule:	$\overline{V} \rightarrow \text{is}$
1st rule:	$\overline{S} \rightarrow \overline{Pr} \ \overline{VP}$																								
2nd rule:	$\overline{NP} \rightarrow \overline{Art} \ \overline{Adj} \ \overline{N}$																								
3rd rule:	$\overline{Adj} \rightarrow \overline{Adv} \ \overline{Adj}$																								
4th rule:	$\overline{VP} \rightarrow \overline{V} \ \overline{NP}$																								
5th rule:	$\overline{Pr} \rightarrow \text{she}$																								
6th rule:	$\overline{Art} \rightarrow \text{a}$																								
7th rule:	$\overline{N} \rightarrow \overline{Adj} \ \overline{N}$																								
8th rule:	$\overline{Adj} \rightarrow \text{pretty}$																								
9th rule:	$\overline{Adj} \rightarrow \text{little}$																								
10th rule:	$\overline{Adv} \rightarrow \text{pretty}$																								
11th rule:	$\overline{N} \rightarrow \text{girl}$																								
12th rule:	$\overline{V} \rightarrow \text{is}$																								

В этой простой грамматике каждое правило имеет форму $g \rightarrow g'$ (читай, « g' заменяет g ») и имеет значение, что g' можно переписать для g в строках символов. Символ S , встречающийся в правилах, можно понимать как обозначающий грамматическую категорию «предложение», Pr - «местоимение», VP - «глагольная фраза», NP - «существительное» и так далее. Символы, отмеченные винкулем (-), составляют множество нетерминальных символов VN . Английские выражения «она» и т. д., встречающиеся в правилах, составляют набор VT терминальных символов. S - начальный символ.

Начиная с S , предложения английского языка могут быть получены в результате применения правил. Вывод начинается с S ; первое

правило позволяет переписать Pr VP для S, получив Pr VP; четвертое правило позволяет переписать V NP для VP, получив Pr Pr NP; и так далее (см. 8). Последний шаг возвращает конечную строку или предложение; он состоит исключительно из элементов терминальной лексики VT. Ни одно из правил не распространяется на него; поэтому дальнейшие шаги невозможны.

- (8) {
- Step (i): \overline{S} , initial symbol
 - Step (ii): $\overline{Pr} \ \overline{VP}$, using 1st rule
 - Step (iii): $\overline{Pr} \ \overline{V} \ \overline{NP}$, using 4th rule
 - Step (iv): $\overline{Pr} \ \overline{V} \ \overline{Art} \ \overline{Adj} \ \overline{N}$, using 2nd rule
 - Step (v): $\overline{Pr} \ \overline{V} \ \overline{Art} \ \overline{Adj} \ \overline{Adj} \ \overline{N}$, using 7th rule
 - Step (vi): she $\overline{V} \ \overline{Art} \ \overline{Adj} \ \overline{Adj} \ \overline{N}$, using 5th rule
 - Step (vii): she is a pretty little girl, using
rules: 12, 6, 8, 9, 11

Множество предложений, генерируемых таким образом грамматикой, называется языком. Помимо тривиальных примеров, грамматики генерируют счетно бесконечные языки.