Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Стоян Михов

Лекция 1: Търсене чрез булеви заявки от ключови думи. Обратен индекс.

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. Въведение в булевото търсене (10 мин)
4. Построяване и на обратен индекс (10 мин)
5. Булево търсене чрез обратен индекс (10 мин)
6. Предварителна обработка на текстовете (10 мин)
7. Позиционен обратен индекс (10 мин)
8. Толериране на близки ключови думи (15 мин)

Логистика на курса

- Лектор: Стоян Михов
  ИИКТ-БАН, бл. 2, стая 322, мейл: stoyan@lml.bas.bg

  - Лекции: Аудитория: ФМИ 404, четвъртък 8:15 — 10:00 часа

- Асистенти: Борис Василев и Румен Михов

  - Упражнения: Аудитория: ФМИ 404, четвъртък 10:15 — 12:00 часа

- Помощни материали: **Moodle** — слайдове, програми, домашни и др.

Какво се надяваме да постигнем

Очаква се студентите да придобият:

1. Познаване на модерните методи за търсене и извличане на информация и обработка на естествен език, както и тяхното реализиране чрез дълбоки невронни мрежи;

2. Начално запознаване с основите на машинното обучение на базата на дълбоки невронни мрежи;

3. Способности за имплементиране на модерни решения в тази област за реализация на реални системи.

Формиране на оценката

- 3 х домашни задания за имплементиране и описание на конкретни задачи:
  3 х 10% = **30%**

- Курсова работа — имплементиране на проект:
  50%

- Устен изпит:
  20%

Учебни материали

- **Основни:**
  - Слайдовете на лекциите, които ще се публикуват в Moodle
  - Кодове на Python от упражненията, които също ще се публикуват в Moodle
- **Допълнителни:**
  - Introduction to Information Retrieval, by C. Manning, P. Raghavan, and H. Schütze (Cambridge University Press, 2008). https://nlp.stanford.edu/IR-book/
  - Neural Network Methods in Natural Language Processing, by Yoav Goldberg and Graeme Hirst. 2017. Morgan & Claypool Publishers.
    линк
  - Deep Learning, by Ian Goodfellow, Yoshua Bengio and Aaron Courville, MIT Press 2016.
    http://www.deeplearningbook.org

План на лекцията

1. За курса (10 мин)
2. **Подходи към проблема за търсене на информация (15 мин)**
3. Въведение в булевото търсене (10 мин)
4. Построяване и на обратен индекс (10 мин)
5. Булево търсене чрез обратен индекс (10 мин)
6. Предварителна обработка на текстовете (10 мин)
7. Позиционен обратен индекс (10 мин)
8. Толериране на близки ключови думи (15 мин)

Задача на търсенето и извличането на информация

- Класическата задача за Търсене на информация (Information Retrieval) се състои в намирането в голяма колекция от неструктурирани текстови документи на документите, които удовлетворяват дадена информационна потребност.

- Класическата задача за Извличането на информация (Information Extraction) се състои в извличане на структурирани елементи (имена, числови стойности, релации и др.) от неструктурирани текстови документи.

- Най-модерният подход, базиран на големи езикови модели (напр. ChatGPT) се състои в отговор на произволни въпроси и задачи формулирани на естествен език, като например:

  - отговор на произволен въпрос, писане на есе, резюмиране на текст, писане на код, извеждане на заключение, и т.н.

- В тази област влизат и множество свързани задачи като:

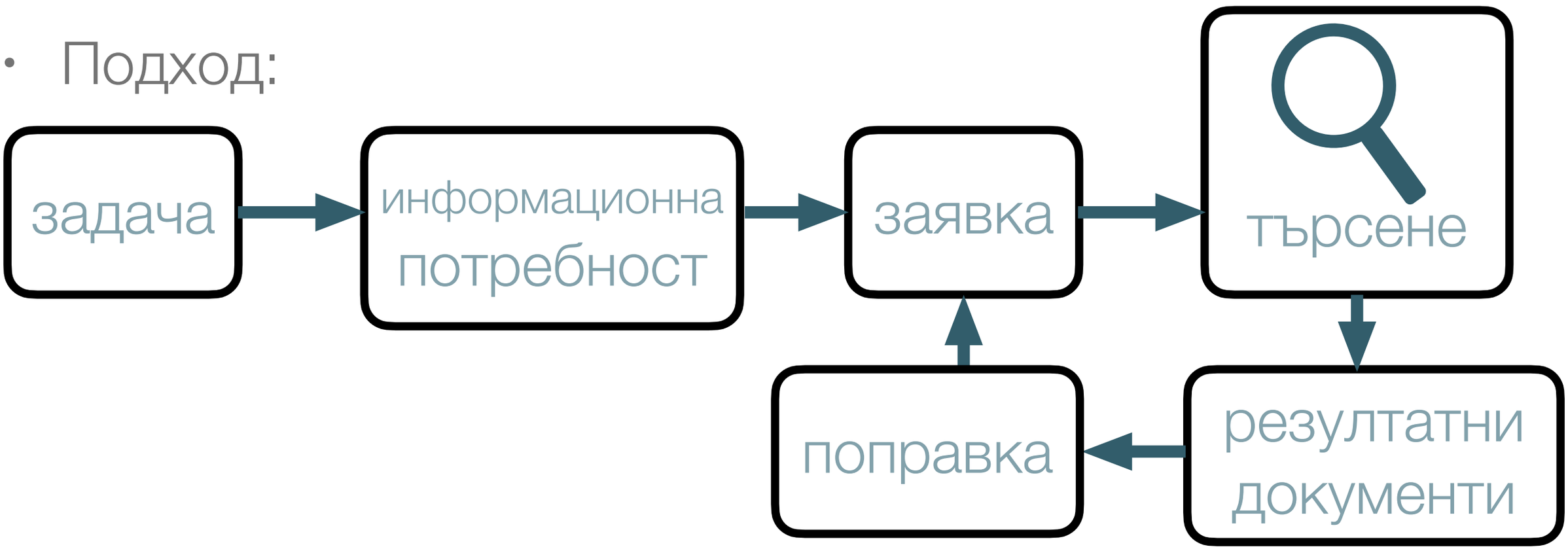
  - класифициране на информация (тематика, сантименталност, значимост),

  - машинен превод,

  - граматичен разбор, и т.н.

Класически подход към търсенето на информация

- Предположения: голяма колекция от документи.
- Цел: намирането на **документите** в колекцията, които са релевантни по отношение на информационната потребност на ползвателя.
- Подход:

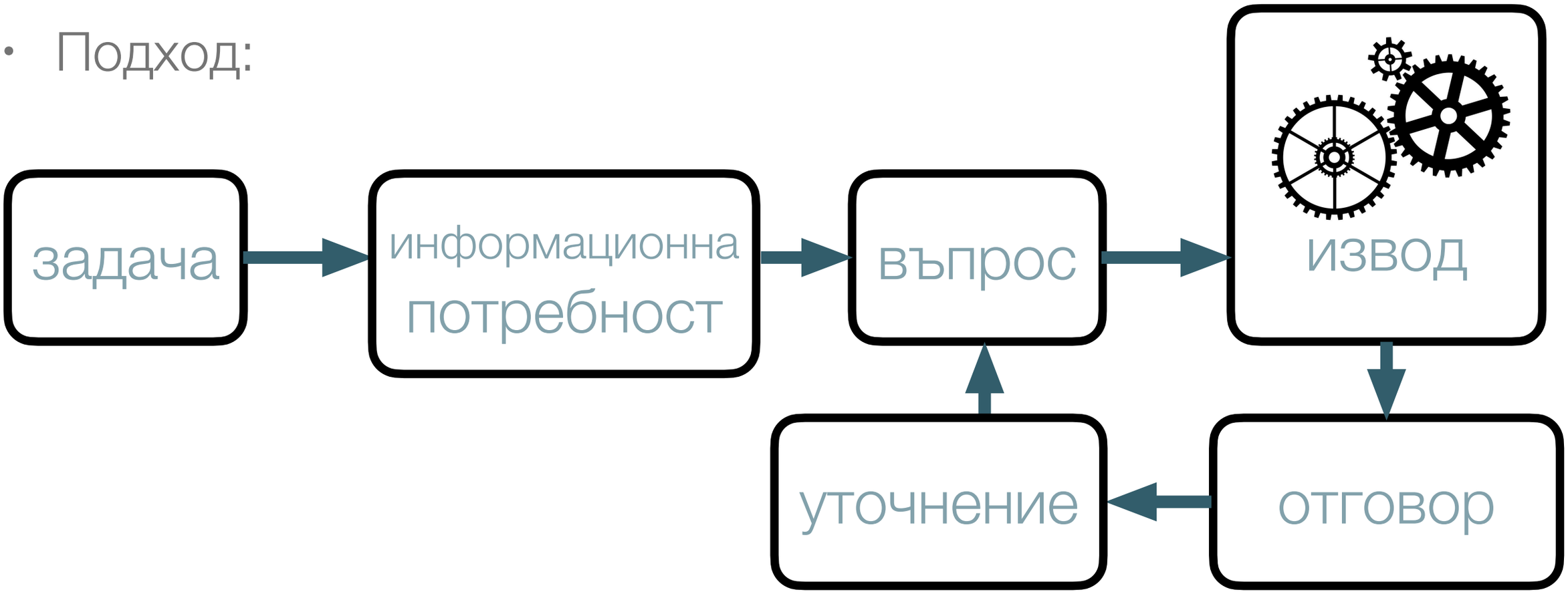


Модерен подход към търсенето и извличането на информация (ще разглеждаме по-нататък)

- Предположения: модел, обучен върху голяма колекция от документи.

- Цел: намирането на **отговора**, които е релевантен по отношение на информационната потребност на ползвателя.

- Подход:



Оценяване на извлечените документи

- **Прецизност**: Процентният дял на релевантните по отношение на потребността документи спрямо всички извлечени.

- **Обхват**: Процентът на извлечените релевантни документи спрямо всички релевантни документи в колекцията.

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. **Въведение в булевото търсене (10 мин)**
4. Построяване и на обратен индекс (10 мин)
5. Булево търсене чрез обратен индекс (10 мин)
6. Предварителна обработка на текстовете (10 мин)
7. Позиционен обратен индекс (10 мин)
8. Толериране на близки ключови думи (15 мин)

Примерна задача

- В кои шекспирови пиеси се срещат героите Цезар и Брут, но не се среща Калпурния?
- Наивен подход: да се използва grep за да се извадят пиесите с Цезар и Брут. След това да се премахнат тези, в които се среща Калпурния
- Проблеми:
  - бавно — трябва да се претърси цялата колекция
  - операцията “НЕ Калпурния“ е проблематична
  - трудно биха се реализирали операции като: намери *Римлянин* в близост до *Сънародник*
  - не позволява ранкирано търсене — да се подредят документите по релевантност

Матрица на срещания
ключова дума / документ

	Антоний и Клеопатра	**Юлий Цезар**	**Бурята**	**Хамлет**	**Отело**	**Макбет**
Антоний	1	1	0	0	0	1
Брут	1	1	0	1	0	0
Цезар	1	1	0	1	1	1
Калпурния	0	1	0	0	0	0
Клеопатра	1	0	0	0	0	0
милост	1	0	1	1	1	1
по-лош	1	0	1	1	1	0
...						

1, ако пиесата съдържа съответната дума, 0 иначе.

Вектори на срещанията

- На всяка ключова дума съпоставяме вектор от 0/1 с размер броя на документите, който отразява срещанията

- За да отговорим на заявката ***Брут* и *Цезар* и не *Калпурния,*** извършваме побитови операции със съответните вектори:

- 110100 &
- 110111 &
- ~010000 =
- **100100**

	Антоний и Клеопатра	Юлий Цезар	Бурята	Хамлет	Отело	Макбет
Антоний	1	1	0	0	0	1
Брут	1	1	0	1	0	0
Цезар	1	1	0	1	1	1
Калпурния	0	1	0	0	0	0
Клеопатра	1	0	0	0	0	0
милост	1	0	1	1	1	1
по-лош	1	0	1	1	1	0
...						

Представяне на матрицата на срещанията

- Нека предположим, че:
  - колекцията съдържа 1 милион документа;
  - средно по 1000 думи в документ;
  - броят на различните думи е 1 милион.
- Матрицата ще съдържа 1000 милиарда елемента по 1 бит получаваме 125TB.
- Броят на единиците е по-малък от 1 милиард — матрицата е силно разредена
- **Решение**: Запаметяваме само позициите на единиците (по-нататък).

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. Въведение в булевото търсене (10 мин)
4. **Построяване и на обратен индекс (10 мин)**
5. Булево търсене чрез обратен индекс (10 мин)
6. Предварителна обработка на текстовете (10 мин)
7. Позиционен обратен индекс (10 мин)
8. Толериране на близки ключови думи (15 мин)

Създаване на списък дума / номер на документ

Документ 1

Антоний във Египет не ще напусне
пиршества и спални за други битки.
Цезар трупа злато, отблъсквайки сърца

Документ 2

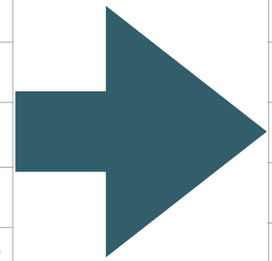
Цезар заслужавал е смъртта си,
Антоний ще обича Цезар мъртъв
не толкоз силно, колкото Брут жив

Дума	Документ
антоний	1
египет	1
напусне	1
пиршества	1
спални	1
други	1
битки	1
цезар	1
трупа	1
злато	1
отблъсквайки	1
сърца	1
цезар	2
заслужавал	2
смъртта	2
антоний	2
обича	2
цезар	2
мъртъв	2
толкоз	2
силно	2
колкото	2
брут	2
жив	2

Сортиране

- Сортираме списъка
  - първо по думите
  - после по документ

Дума	Документ
антоний	1
египет	1
напусне	1
пиршества	1
спални	1
други	1
битки	1
цезар	1
трупа	1
злато	1
отблъсквайки	1
сърца	1
цезар	2
заслужавал	2
смъртта	2
антоний	2
обича	2
цезар	2
мъртъв	2
толкоз	2
силно	2
колкото	2
брут	2
жив	2

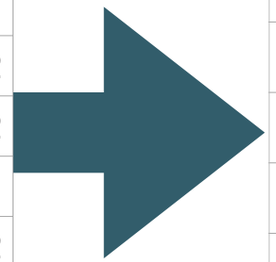


Дума	Документ
антоний	1
антоний	2
битки	1
брут	2
други	1
египет	1
жив	2
заслужавал	2
злато	1
колкото	2
мъртъв	2
напусне	1
обича	2
отблъсквайки	1
пиршества	1
силно	2
смъртта	2
спални	1
сърца	1
толкоз	2
трупа	1
цезар	1
цезар	2
цезар	2

Създаване на обратен индекс

- Обединяваме повторенията
- Разделяме речника от срещанията
- Добавяме поле за честота (броя) на срещанията

Дума	Документ
антоний	1
антоний	2
битки	1
брут	2
други	1
египет	1
жив	2
заслужавал	2
злато	1
колкото	2
мъртъв	2
напусне	1
обича	2
отблъсквайки	1
пиршества	1
силно	2
смъртта	2
спални	1
сърца	1
толкоз	2
трупа	1
цезар	1
цезар	2
цезар	2



Речник		Списък срещания
Дума	Честота	
антоний	2	→ 1 → 2
битки	1	→ 1
брут	1	→ 2
други	1	→ 1
египет	1	→ 1
жив	1	→ 2
заслужавал	1	→ 2
злато	1	→ 1
колкото	1	→ 2
мъртъв	1	→ 2
напусне	1	→ 1
обича	1	→ 2
отблъсквайки	1	→ 1
пиршества	1	→ 1
силно	1	→ 2
смъртта	1	→ 2
спални	1	→ 1
сърца	1	→ 1
толкоз	1	→ 2
трупа	1	→ 1
цезар	2	→ 1 → 2

Резултат

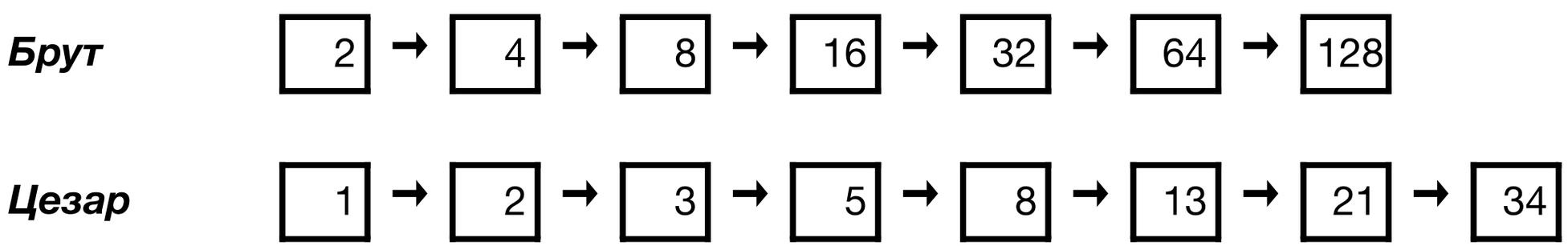
- Време за построяване на обратен индекс?
- Памет за построяване на обратен индекс?

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. Въведение в булевото търсене (10 мин)
4. Построяване и на обратен индекс (10 мин)
5. **Булево търсене чрез обратен индекс (10 мин)**
6. Предварителна обработка на текстовете (10 мин)
7. Позиционен обратен индекс (10 мин)
8. Толериране на близки ключови думи (15 мин)

Реализиране на конюнкция: сливане на срещанията

- Разглеждаме заявката ***Брут И Цезар***
- Извличаме от речника списъците от срещания за *Брут* и *Цезар*
- “Сливаме” двата списъка като направим тяхното сечение
- Обхождаме синхронно двата списъка за време линейно спрямо сумата от дължините им
- Ако дължините на списъците са съответно ***x*** и ***y***, то сливането отнема време ***O(x+y)***
- **ВАЖНО:** Списъците трябва да са сортирани по номер на документ



Алгоритъм за сечение на два списъка от срещания (сливане на списъци)

```
Intersect(p1,p2)
1  answer <- <>
2  while p1 != NIL and p2 != NIL do
3     if docID(p1) == docID(p2) then
4        Add(answer,docID(p1))
5        p1 <- next(p1)
6        p2 <- next(p2)
7     else if docID(p1) < docID(p2) then
8        p1 <- next(p1)
9     else
10       p2 <- next(p2)
11 return answer
```

Алгоритми за сливане на списъци за други булеви заявки

- Дизюнкция: ***X или Y***
- Конюнкция с отрицание: ***X и не Y***
- Дизюнкция с отрицание: ***X или не Y***

- Сложност на съответните алгоритми.
- Може ли винаги да сливаме за време ***O(x+y)***? Какво можем да постигнем?

Алгоритъм за произволни булеви заявки

- Разбиваме сложен израз на подизрази:
  (Брут или Цезар) и (Антоний или не Клеопатра)

- Сливаме всеки от подизразите и получаваме списък от срещания, след което сливаме получените списъци до получаването на срещанията за цялата заявка:

Брут или Цезар -> A
Антоний или не Клеопатра -> B
A и B -> C

Оптимизация на изпълнението на заявката

- Нека е дадена заявка, която е конюнкция на *n* терма
- За всеки от термовете получаваме списъка от срещанията и съответните им дължини
- В какъв ред да ги слеем?

Оптимизация на изпълнението на заявката

- Нека е дадена заявка, която е конюнкция на *n* терма
- За всеки от термовете получаваме списъка от срещанията и съответните им дължини
- В какъв ред да ги слеем?
- Отговор:
  - започваме с най-късите списъци и сливаме в нарастващ ред.

По-обща оптимизация

- За единичните термове речникът ни дава броя на срещанията им
- Оценяваме броя на срещанията на конюнкцията с минимума на броя на срещанията на съответните списъци на конюнктите
- Оценяваме броя на срещанията на дизюнкцията със сумата на броя на срещанията на съответните списъци на дизюнктите
- Извършваме сливанията в нарастващ ред
- **Задача:** Каква сложност може да постигнем в най-общия случай — при наличие на отрицания.

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. Въведение в булевото търсене (10 мин)
4. Построяване и на обратен индекс (10 мин)
5. Булево търсене чрез обратен индекс (10 мин)
6. **Предварителна обработка на текстовете (10 мин)**
7. Позиционен обратен индекс (10 мин)
8. Толериране на близки ключови думи (15 мин)

Етапи на предварителната обработка на текста

- Разбиване текста на единици (Tokenization)
  - изрази като: *25-милиметров, полу-слято, 01.10.2020, г’син, …*
- Нормализация
  - главни / малки букви, съкращения, акроними: *ТЕКСТ, др., С.У., …*
- Преобразуване до основни форми (Stemming)
  - дунавски →дунав, вървейки →вървя, …
- Премахване на ***стоп*** думи
  - съюзи, предлози, междуметия, кратки местоимения, спомагателни глаголи, които са много често срещани, но нямат собствено значение

Технологии и инструменти за предварителна обработка на текст

- Най-често базирани на експертни правила и речници
  - използване на регулярни изрази и релации
  - използване на речници за съкращения, акроними, …
- правила за преобразуване и замяна
- използване на крайни автомати и преобразуватели
  - курс летен семестър: *Приложения на крайните автомати (ПКА)*

Реализация на речник

- Ефективно представяне на ключовите думи
  - Бързо търсене
  - Възможност за поддържане — добавяне и триене
  - Сбито представяне
- Технологии
  - хешове, балансирани дървета, перфектни хешове, крайни автомати (ПКА)

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. Въведение в булевото търсене (10 мин)
4. Построяване и на обратен индекс (10 мин)
5. Булево търсене чрез обратен индекс (10 мин)
6. Предварителна обработка на текстовете (10 мин)
7. **Позиционен обратен индекс (10 мин)**
8. Толериране на близки ключови думи (15 мин)

Съставни имена, фрази и изрази

- Примери: Иван Иванов, Стара Загора, операционна система, така нататък
- Първо решение — добавяне на двойки от думи към речника
- Огромно нарастване на речника и списъците от срещания
- При нужда от намиране на тройки или четворки е необходимо разбиване на двойки и проверка дали са последователни
- Има смисъл само за често срещани съставни имена и изрази

Алтернативно решение: използване на позиционен индекс

- Към списъка от срещания за всяко срещане на терм в документ добавяме списък от позициите в документа на съответните срещания

- Пример: **да бъде или да не бъде**

```
да:
   2 -> 1,17,74,222,551;
   4 -> 8,16,190,429,433;
   7 -> 13,23,191;
   ...
```

```
бъде:
   1 -> 17,19;
   4 -> 17,191,291,430,434;
   5 -> 14,19,101;
   ...
```

Търсене с помощта на позиционен индекс

- Позволява в булевата заявка да се добави ограничение за разстоянието между термовете:
  - да и/1 бъде
  - компютърна и/3 мрежа
- Изисква съществено допълване на алгоритмите за сливане
- Изисква между 2 и 4 пъти повече памет за представяне на индексите.

Алгоритъм за позиционно сечение на два списъка от срещания

```
PositionalIntersection(p1, p2, k)
1   answer <- <>
2   while p1 != NIL and p2 != NIL do
3       if docID(p1) == docID(p2) then
4           l <- <>
5           pp1 <- positions(p1)
6           pp2 <- positions(p2)
7           while pp1 != NIL do
8               while pp2 != NIL do
9                   if |pos(pp1) - pos(pp2)| <= k then
10                      ADD(l, pos(pp2))
11                  else if pos(pp2) > pos(pp1) then
12                      break
13                  pp2 <- next(pp2)
14              while l != <> and |l[0] - pos(pp1)| > k do
15                  DELETE(l[0])
16              for each ps in l do
17                  ADD(answer, <docID(p1), pos(pp1), ps>)
18              pp1 <- next(pp1)
19          p1 <- next(p1)
20          p2 <- next(p2)
21      else if docID(p1) < docID(p2) then
22          p1 <- next(p1)
23      else
24          p2 <- next(p2)
25  return answer
```

План на лекцията

1. За курса (10 мин)
2. Подходи към проблема за търсене на информация (15 мин)
3. Въведение в булевото търсене (10 мин)
4. Построяване и на обратен индекс (10 мин)
5. Булево търсене чрез обратен индекс (10 мин)
6. Предварителна обработка на текстовете (10 мин)
7. Позиционен обратен индекс (10 мин)
8. **Толериране на близки ключови думи (15 мин)**

Стриктно търсене по зададени ключови думи

Пропускане на релевантни документи поради това, че:

- Не се обхващат различни форми на ключовата дума: компютрите, вятърните, вдървяването, …
- Не се толерират правописни грешки: невроната, ябалка, инжинер, …
- Различие в изписването, както в заявката, така и в документа.

Решение: Толериране на близки по изписване ключови думи

Толериране на форми на дадена дума

- Чрез използване на stemming:
  - позиционен -> позиция, дунавската -> дунав, …
  - втори речник върху “основите” на думите, за всяка основа се посочва списък от ключови думи със съответната основа
- Води до повече резултати, в някои случай - нерелевантни:
  - хлебарка -> хляб, националистически -> национален -> национал, …
- Не се толерират грешки в изписването

Използване на шаблони (wildcard)

- Използване на * за означаване на произволен подниз в ключовата дума:
  позиц*, компют*, по*ните, *ход*

- Реализация на * в края на заявката чрез обхождане на поддървото (подавтомата) на речник, представен с дърво или автомат, от върха, който се достига с дадения префикс

- Реализация на * в средата на заявката чрез обхождане на поддървото (подавтомата) на речник, представящ пермутациите на думите
  ма*на -> ма*на$ -> на$ма*

  машина$
  ашина$м
  шина$ма
  ина$маш
  на$маши
  а$машин

Разстояние на корекция (Edit distance)
Разстояние на Левенщайн

- Минималния брой елементарни корекции (изтриване, вмъкване и заменяне на единични символи), необходими за преобразуване на един низ до друг.

- Индуктивна схема:

$$d_L(\varepsilon, W) = |W|$$

$$d_L(P, \varepsilon) = |P|$$

$$d_L(Pa, Wb) = \begin{cases} d_L(P, W) & \text{if } a = b \\ 1 + \min(d_L(P, W), d_L(Pa, W), d_L(P, Wb)) & \text{if } a \neq b \end{cases}$$

- Модификации на разстоянието на корекция: добавяне на транспозиция, сливане и разбиване на символи, тегла на корекциите

Алгоритъм за намиране на Левенщайн разстояние между два низа

```
EditDistance(s1, s2)
1   int m[i, j]=0
2   for i <- 1 to |s1| do
3       m[i, 0]=i
4   for j <- 1 to |s2| do
5       m[0, j]=j
6   for i <- 1 to |s1| do
7       for j <- 1 to |s2| do
8           m[i, j] = min{m[i - 1, j - 1] +
9                             if s1[i] == s2[j] then 0 else 1,
10                        m[i - 1, j] + 1,
11                        m[i, j - 1] + 1}
12 return m[|s1|,|s2|]
```

Интегриране на толерантно търсене в общата схема на булевото търсене

- За всяка от ключовите думи/шаблони от заявката се намира съответен списък от “близки” думи, срещани в документите.

- Списъците от срещания за дадена ключова дума от заявката се обединяват.

- Резултатът се формира като се извършат останалите булеви операции с така обединените списъци на срещания на термовете.

Заключение

- Класическите системи за търсене на информация използват техники базирани на обратни индекси с цел ефективност.

- Много от професионалистите търсещи информация в специализирани бази (например в юридически или научни документи) предпочитат използването на разширено булево търсене.

- В по-модерните потребителски системи напоследък се предпочита търсене по “смисъл” и ранкиране на резултатите (ще бъде разгледано в следващите лекции).