**Инфопоиск, контрольная №2**

# 1. Автоматическая классификация текстов. Задачи классификации. Основные подходы к автоматической классификации (рубрицирования) текстов.

## Классификация текстов

Виды:

1. Тематическая классификация (рубрикация)
2. Жанровая классификация (новости – научные тексты – техническое задание)
3. Классификация по авторам
4. Фильтрация спама
5. Классификация по тональности текстов

Должна быть задана система классов (категорий) для классификации vs. Кластеризация

**Классификация/рубрикация информации** – отнесение порции информации к одной или нескольким категориям из конечного множества рубрик.

**(ВИКИПЕДИЯ) Задача классифика́ции** — формализованная **задача**, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой.

## Методы рубрицирования текстов:

1. Ручное рубрицирование
2. Автоматическое
   1. Инженерный подход (=методы, основанные на знаниях, экспертные методы)
   2. Методы машинного обучения
3. Полуавтоматическое

**Методы *машинного обучения* для задачи автоматической рубрикации**

1. Метод Байеса (Naive Bayes )
2. Метод Roccio
3. Метод ближайшего соседа (k-Nearest Neighbors – knn)
4. Метод опорных векторов (Support-vector machines – SVM)
5. !!Должно быть размечено много данных.

# 2. Что такое инженерный метод классификации текстов? Плюсы и минусы инженерных методов классификации

## Автоматическая рубрикация: Инженерный подход

**Основное предположение**: рубрикатор создается осмысленно, содержание рубрики можно выразить ограниченным количеством понятий в виде формулы.

Эксперты описывают смысл рубрики в виде булевских выражений, правил продукции.

Плюсы:

* 1. высокая эффективность
  2. «прозрачность» получаемых результатов

Минусы:

* трудоемкость описания рубрик

## + Проблемы методов, основанных на знаниях

* Содержание рубрики сложнее, чем это выглядит по формулировке
* Лексическая многозначность
* Ложная корреляция
* Нестандартный контекст употребления терминов
* Упоминание терминов вне главной темы
* Неполнота описания рубрики

## !!!! Проблемы рубрикатора:

1. появление лишних рубрик
2. пропуск нужной рубрики

# 3. Укажите плюсы и минусы ручного рубрицирования.

## Ручное рубрицирование

1. Высокая точность рубрицирования
   1. Обычно процент документов, в которых проставлена явно неправильная рубрика, чрезвычайно мал
2. Низкая полнота рубрицирования
   * одна-две основных рубрики, характеризующие основное содержание документа, хотя документ может быть отнесен и к ряду других рубрик.
   * В результате получается, что
     1. Процент совпадения результатов рубрицирования различных экспертов весьма низкий - 60 %.
     2. В результате похожие документы могут получить достаточно разные наборы рубрик
     3. Непоследовательность ручного рубрицирования
     4. Низкая скорость обработки документов

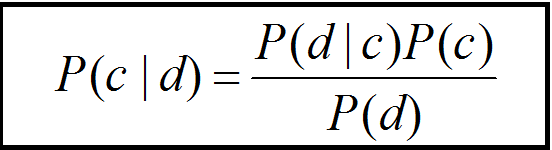
# 4. Метод Байеса для автоматической классификации текстов

1. Пользователь (эксперт) – размечает документы, относящиеся к рубрике

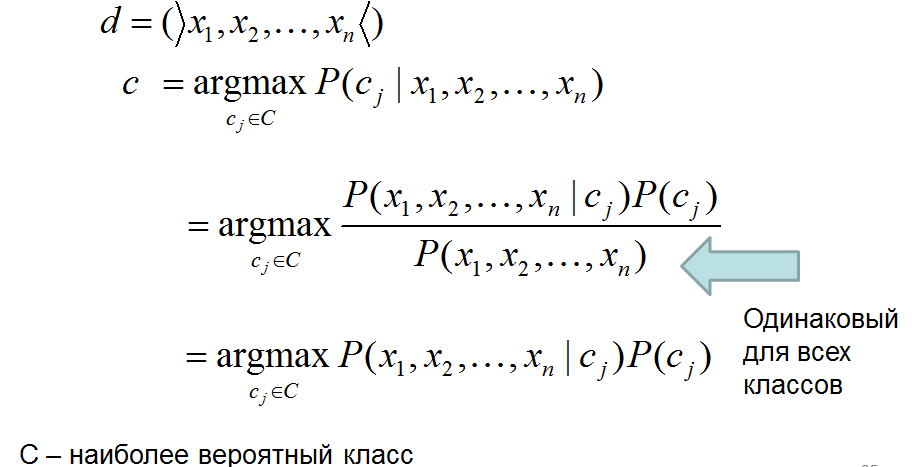
* Эта разметка производится оффлайн.
* Может быть результатом ручной рубрикации

1. Определение лучшего класса для документа

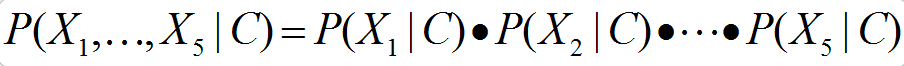
* P (C|di) – условная вероятность: вероятность класса С при предъявлении документа di
* Метод Байеса из курса теории вероятности



Задача: Классифицировать новый документ *d на основе совокупности признаков в один из классов cj* ∈ *C*



**Предположение о Conditional Independence Assumption:** признаки присутствия термов независимы друг от друга в заданном классе классификатора:



**Обучение модели: вычисление нужных величин**

Первая попытка: частотная оценка. Используем просто частоты в обучающих примерах

* + - 1. Проблема: Отсутствие одного признака в обучающей выборке делает вероятность нулевой!? => Сглаживание: добавляем тем признакам, которые не встречались, немного вероятности.

Базовый подход к классификации текстов на основе метода Байеса:

1. Атрибуты - позиции в тексте, значения - слова
2. Предположим еще, что классификация не зависит от позиции слов

* Используем те же параметры в любой позиции
* Результат – модель мешка слов

# 5. Байесовский классификатор в задаче обнаружения поискового спама

**Спам** - рассылка коммерческой и иной рекламы или иных видов сообщений лицам, не выражавшим желания их получать.

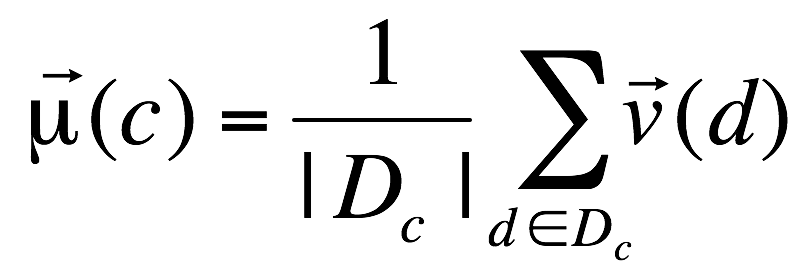
**Вычисление признаков**:

1. Заголовок: получатель, отправитель, доменные имена
2. Текст
   1. Слова, фразы, строки символов
   2. Могут быть бинарными или числовыми
   3. URL, HTML tags, картинки

**Байес для распознавания спама**

1. Два класса: спам – неспам
2. Вычисление вероятностей
   * Вероятности классов: доля каждого класса в обучающей выборке
   * *Вероятности признаков P*(*xk | cj*): *– количество вхождений в классе/число признаков в классе*
3. Могут быть варианты формул, связанные с тем, что пропустить спам менее опасно, чем отправить нормальное письмо в спам
4. Используется во многих системах фильтрации и в настоящее время
   * Возможно, что предположение о независимости признаков, больше соответствует действительности

# 6. Метод Роккио для автоматической классификации текстов

Определение центроида

где *Dc* is множество всех документов, которые принадлежат классу *c* and *v*(*d*) is векторное представление документа *d.*

Выводы:

* Простое представление для каждого класса: центроид-прототип
* Классификация основана на близости центроиду
* Мало используется вне классификации текстов
  + Может быть эффективна для классификации текстов
  + Но обычно хуже, чем Naïve Bayes
* Простое обучение и тестирование

# 7. Метод Knn для автоматической классификации текстов

* kNN = k Nearest Neighbor
* Чтобы классифицировать документ *d* в класс c нужно:
  + определить *k*-окружение as *k* ближайших соседей *d*
  + Посчитать число документов в N, которые относятся к c
  + Оценить P(c|*d*) as i/k
  + Выбрать максимальный класс argmaxc P(c|*d*) [ = majority class]

# 8. Поясните основной принцип метода SVM для автоматической классификации текстов

* Метод опорных векторов (SVM) находит оптимальное решение
  + Максимизирует расстояние между гиперплоскостью и трудными точками, близкими к границе раздела
  + Интуитивно: если нет точек около границы раздела, то нет и сложных (неопределенных) примеров
* Предполагается, что точки имеют вид {(x1,c1),… (xn,cn)}, где *ci* принимает значение 1 или −1, в зависимости от того, какому классу принадлежит точка
* Каждое xi - это p-мерный вещественный вектор, обычно нормализованный значениями [0,1] или [-1,1].
* Нужно найти разделяющую гиперплоскость, которая имеет вид:
* w\*x-b=0, где w – перпендикуляр к разделяющей гиперплоскости

Формула: Классифицирующая линейная функция

f(x) = Σαici<xi,x> + b

где xi – это опорные вектора

<xi,x> - скалярное произведение опорного вектора на вектор текущего документа

сi – это {-1,1}

αi – вес опорных векторов

# 9. Плюсы и минусы методов машинного обучения для классификации текстов

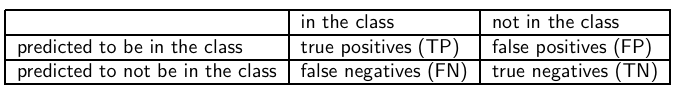
* Машинное обучение
  + Эффективно при наличии качественно размеченной обучающей коллекции
  + низкая эффективность при большом числе рубрик
  + трудно интерпретируемые результаты («черный ящик»)

# 10. Особенности применения методов машинного обучения при классификации текстов в зависимости от размера обучающей коллекции

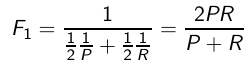
* Нет данных для обучения — правила пишутся вручную
* Мало данных — классификация «с учителем» + нужно получить размеченные данные по возможности быстрее
* Достаточное количество данных — подходит для пррименения SVM можно использовать булевские модели
* Большое количество данных — хорошо подходит в теории, то SVM и KNN непрактичны, можно использовать Naive Bayes

# 11. Метрики качества для систем автоматической классификации (рубрикации) текстов. Микро и макро усреднение

**Точность *P* и полнота *R***



*P=TP* / ( *TP* + *FP*)*, R* = *TP* / ( *TP* + *FN*)



Комбинированная мера: *F*

**Усреднение: Micro vs. Macro**

* Посчитали меру оценки (*F*1) for one class.
* Как агрегировать оценки F1 для многих классов.
* Macroaveraging - макроусреднение
  + Посчитать *F*1 для каждого из *C* классов
  + Среднее арифметическое для этих *C* чисел
* Microaveraging - микроусреднение
  + Посчитать TP, FP, FN для каждого из *C* classes
  + Суммировать эти С чисел для каждого показателя
  + Посчитать *F*1 для суммированных TP, FP, FN

# 12. Что такое кластеризация текстов? Чем она отличается от классификации (рубрикации) текстов? Типы методов кластеризации

**Алгоритмы кластеризации**

**А**втоматическая кластеризация текстов**:**

* **И**меется текстовая коллекция
* Нужно разбить коллекцию на классы близких документов
* Могут быть созданы иерархические классы
* Сейчас: одно из важных средств для визуализации большой выдачи документов при поиске
* Для визуализации важно: хорошее название кластера
* Примеры:
  + Новостные агрегаторы (Яндекс.Новости, Рамблер.Новости, Google.News, Новотека)
  + Кластеризация результатов поиска (Clusty, Нигма)

Плоские алгоритмы

1. Обычно начинаются со случайного разбиения
2. Итеративное уточнение

*K средних (К* means)

Иерархические алгоритмы

* Снизу-вверх, аггломеративный
* (Сверху-вниз, разбиение)

**Жесткая и мягкая кластеризация**

* Жесткая кластеризация: Каждый документ принадлежит только к одному кластеру
  + Легче выполнить
* Мягкая кластеризация: Документ может принадлежать более, чем к одному кластеру
  + Полезно, но сложнее разбивать и использовать

# 13. Метод K-means для кластеризации текстов

*K*-Means: основные идеи

* Рассматривает документы как вектора с вещественными значениями
* Кластеры базируются на понятии центроида точек в кластере , *c* :

**Ошибка! Внедренный объект недопустим.**

* Присваивание документов к кластеру базируется на сходстве с теущими центроидами кластеров

*Алгоритм K*-Means

* Выберем *K* случайных документов {*s*1, *s*2,… *sK*} как исходное множество (seeds) – это как бы центроиды будущих кластеров.
* До тех пор пока кластеризация не сойдется (или другой критерий остановки):

Для каждого документа *di*:

Присваиваем *di* к кластеру *cj* такому, что similarity(*xi*, *s*j) - максимально.

Затем обновляем множество {*s*1, *s*2,… *sK*}, заменяем на центроиды текущих кластеров.

Для каждого кластера *cj:*

*s*j = μ(*cj*)

Условия остановки

* Несколько возможностей
  + Фиксированное число итераций
  + Не меняется разделение по документов
  + Не меняется позиция центроидов

Сколько кластеров?

* Число кластеров задается – *K* 
  + Разделяет документы на определенное число кластеров
* Нахождение правильного числа кластеров – это часть проблемы
* Могут использоваться специальные методы подбора k

Особенности:

-- классический метод – задание фиксированного множества кластеров

-- кластеры стремятся быть одинаковыми и «круглыми»

Итог: проблемы K-means

* K-means работает, если
  + Кластеры сферические
  + Кластеры хорошо разделены
  + Кластеры похожего объемы
  + Кластеры имеют сходное количество элементов

# 14. Аггломеративная кластеризация – основной принцип и подвиды

Иерархическая аггломеративная кластеризация

* Начинает с рассмотрения документов как отдельных кластеров
  + итеративно объединяет ближайшую пару кластеров, до тех пор пока не останется один кластер.
* История объединения и образует бинарное дерево или иерархию

Ближайшая пара кластеров

* Много способов определения, что такое наиболее сходная пара кластеров
* **Single-link**
  + Сходство по наиболее похожим документам (single-link)
* **Complete-link**
  + Сходство по наиболее непохожим документам
* **Центроид**
  + Сходство по наиболее похожим центроидам
* **Average-link**
  + Средний косинус между парами элементов двух кластеров

Аггломеративная кластеризация: Single Link

* Использует максимальное сходство пар:

**Ошибка! Внедренный объект недопустим.**

* Может породить длинные и тонкие кластеры - цепочки.
* После склеивания *ci* и *cj*, сходство результирующего кластера к другому кластеру, *ck*,:

**Ошибка! Внедренный объект недопустим.**

Кластеризация по всем связям (complete link)

* Использует наименее сходные пары:

**Ошибка! Внедренный объект недопустим.**

* Создает более «плотные», сферические кластеры.
* После склеивания *ci* и *cj*, сходство результирующего кластера с другим кластером, *ck*, :

**Ошибка! Внедренный объект недопустим.**

# 15. Методы тестирования автоматической кластеризации

Что такое хорошая кластеризация?

* Внутренний критерий: Хорошая кластеризация производит качественные кластеры, в которых:
  + Внутри кластера сходство высокое
  + Между классами – сходство низкое
  + Измеряемое качество кластеризации зависит и от документа, и от меры сходства

Внешние критерии качества кластеризации

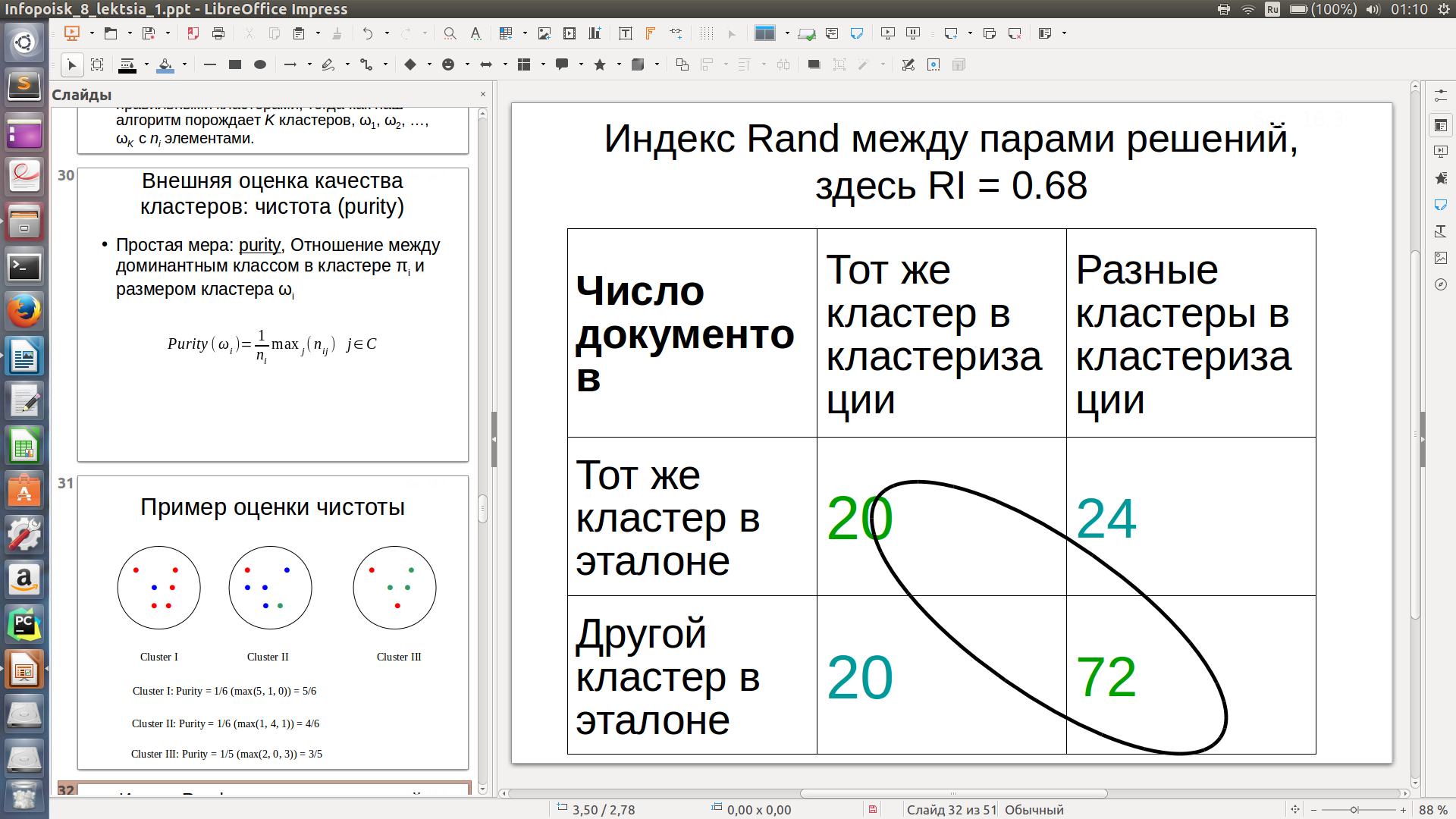
* Качество измеряет способность кластеризации обнаруживать скрытые классы объектов в эталонных данных (gold standard)
* Оценивает кластеризацию по отношению к «истинным» кластерам (ground truth) … требует *размеченных данных*
* Предположим, что имеются документы с *C* правильными кластерами, тогда как наш алгоритм порождает *K* кластеров, ω1, ω2, …, ω*K*  с *ni* элементами.

Внешняя оценка качества кластеров: чистота (purity)

* Простая мера : purity , Отношение между доминантным классом в кластере πi и размером кластера ωi

**Ошибка! Внедренный объект недопустим.**

Rand index и F-мера



**Ошибка! Внедренный объект недопустим.**

Сравним со стандартными полнотой и точностью:



Возможно, лучшей мерой является F-мера

16. Особенности кластеризации потока новостей в реальном времени

# Особенности обработки новостного потока

* Корпус документов постоянно пополняется
* Временное окно (24-72-120 часов)
* Разные размеры
* Наличие дубликатов, определение первоисточника
* Ошибки при сборе новостных сообщений:
  + ошибки очистки
  + ошибки датировки
* Спамерские технологии источников

Новость: треугольник: главное – предыстория – обсуждение, комментарии

# Требования к кластеризации

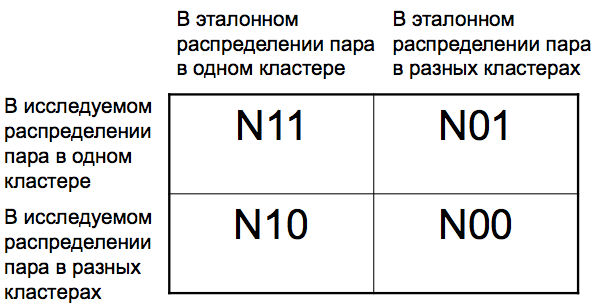
* Минимизация времени работы (максимально 15-30 минут)
* Актуальность (главное сообщение)
* Публикация всех кластеров(не только больших, но и малых)
* Учет перепечаток
* Точность важнее полноты
* Эволюционность кластеризации
* Учет ручного вмешательства:
  + корректировка кластеризации
  + корректировка представления на портале

# Эталонное распределение

Возможности редактора:

1. Визуализировать, сортировать кластеры по дате, близости к центру и т.п.
2. Просматривать близкие кластеры к рассматриваемому (кандидаты на склейку)
3. Объединять близкие кластеры
4. Разделять существующие кластеры

# Внутренние меры



Точность: P = N11 / (N11 + N01)

Полнота: R = N11 / (N11 + N10)

F1-мера: F1 = 2 \* P \* R / (P + R)

17. Автоматическое аннотирование. Виды автоматических аннотаций.

Автоматическое аннотирование документа (совокупности близких по смыслу документов) - автоматическая технология, передающая в краткой форме основное содержание документа (совокупности документов)

* Аннотирование отдельного документа,
* Аннотирование совокупности документов -построение обзорного реферата

Назначение: быстрое ознакомление с содержанием документа (совокупности документов)

# Типы аннотаций: Абстракты vs. Экстракты

* Экстракты получаются извлечением фрагментов исходных текстов (обычно предложений) – основная применяемая технология
* Абстракты порождаются – экспериментальные технологии

# Типы по основному содержанию

* Индикативные – собственно содержание
* Информативные – упор на цифры, данные
* Оценочные – упор на мнения

# Типы аннотаций по содержанию

* Индикативная:

Авария произошла накануне на юго-востоке Москвы: экипаж отдела вневедомственной охраны двигался на служебном автомобиле по Волжскому бульвару и сбил молодого человека, который переходил дорогу с велосипедом на зеленый свет.

* Оценочная:

Ужасная по своей нелепости трагедия произошла в доме на Митинской улице. Маленькая девочка утонула... в аквариуме!

# По фокусу

* Общее содержание
* В ответ на запрос (Query-based - сниппет) – контекстная аннотация, тематически-ориентированная аннотация

# По структуре

* Связная аннотация
* Аннотация ключевыми словами

# Одного документа или многих документов

# На том же языке, на другом языке**18. Методы и признаки для отбора предложений в экстрактивном методе автоматического аннотирования**

# Подходы к формированию экстрактов для многих документов

* Частотные подходы: (самое важное должно быть частотно)

А) Метод SumBasic: наиболее частотные слова с большей вероятностью должны оказаться в аннотации

* На каждой итерации происходит расчет вероятностей слов => предложение с max средней вероятностей слов
* Пересчет вероятностей для слов из отобранного предложения => снижение повторов

В) Использование tf-idf (tf – в исходном документе, idf – в корпусе, из которого извлечен документ)

* Подходы, основанные на методах оптимизации

А) локальная оптимизация – оптимизация выбора следующего предложения

В) глобальная оптимизация – оптимизация некотрой функции

С) MMR

* Подходы, основанные на графах (Центральность предложения)

Вершины – предложения, дуги – сходство

* Подходы, основанные на машинном обучении (Признаки предложения и их комбинирование)

# Характеристики для извлечения предложения

* Частотные слова текста
* Учет заголовка
* Присутствие ключевых слов и конструкций
* Позиция в тексте
* Связность с предыдущим
* Новизна информации

19. Метод MMR автоматического аннотирования

* (Carbonell, Goldstein, 1998)
* Итеративный метод – локальная оптимизация
* На каждой итерации производится ранжирование предложений-кандидатов
* В итоговую аннотацию отбирается одно с самым высоким рангом
* Максимизировать сходство с исходным документов (набором документов)
* Минимизировать сходство с уже отобранными в аннотацию предложениями

1. При использовании в контекстно-зависимой аннотации – максимизируется сходство с запросом

Пусть:

1. Q – запрос к системе
2. S – множество предложений кандидатов
3. s – рассматриваемое предложение кандидат
4. Е – множество выбранных предложений

Тогда:



# Пример аннотации (100 слов) по многим документам (Метод MMR)

Президент "АЛРОСА" Александр Ничипорук уходит со своего поста.

1. Восьми улусам Якутии принадлежит 8 % акций компании.

2. Отношения между нынешним и бывшим президентами АЛРОСА (господин Штыров возглавлял компанию до того как стал президентом Якутии) не сложились с момента прихода господина Ничипорука два года назад.

3. Александр Ничипорук покидает пост президента компании АЛРОСА, на которую приходится четверть мировой добычи алмазов.

4. Росимуществу принадлежит 37 % акций АЛРОСА, минимуществу Якутии - 32 %, восьми улусам Якутии - 8 %, физическим и юридическим лицам - 23 %, из которых ВТБ владеет 10, 5 % акций.

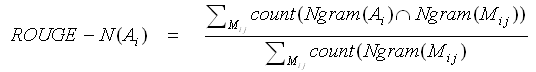
5. Напомним, что до Александра Ничипорука АК АЛРОСА возглавлял Владимир Калитин (март 2002 - декабрь 2004 года), а еще ранее - действующий президент Якутии Вячеслав Штыров (1996-2002 годы).

20. Метрика Rouge для тестирования автоматических аннотаций

# Автоматические ROUGE метрики

* **ROUGE** или **Recall-Oriented Understudy for Gisting Evaluation** – набор метрик и комплекс программ для оценки автоматического аннотирования и машинного перевода текстов.
* Основная идея – сравнение генерированного текста с “эталонным”, сделанным человеком.
* Существуют различные формы метрики, сравнивающие:
  + n-граммы (ROUGE-N)
  + минимальные общие подстроки (ROUGE-L и ROUGE-W)
  + униграммы и биграммы (ROUGE-1 and ROUGE-2)

# Общая формула:



* *Ai* – оцениваемая обзорная аннотация *i*-того кластера.
* *Mij* – ручные аннотации *i*‑того кластера.
* *Ngram(D)* – множество всех n-грамм из лемм соответствующего документа *D.*

# Пример Rouge для двух предложений:

* Китай и Тайвань установили авиасообщение после 60-летнего перерыва.
* После почти 60-летнего перерыва открылось регулярное авиасообщение между Тайванем и материковым Китаем.

Rouge-1 = 7/12 = 0.58

# Оценка ROUGE:

+ Малое участие человека, лёгкость применения

- Отсутствие оценки читабельности, результат не всегда идеален с точки зрения человека

21.

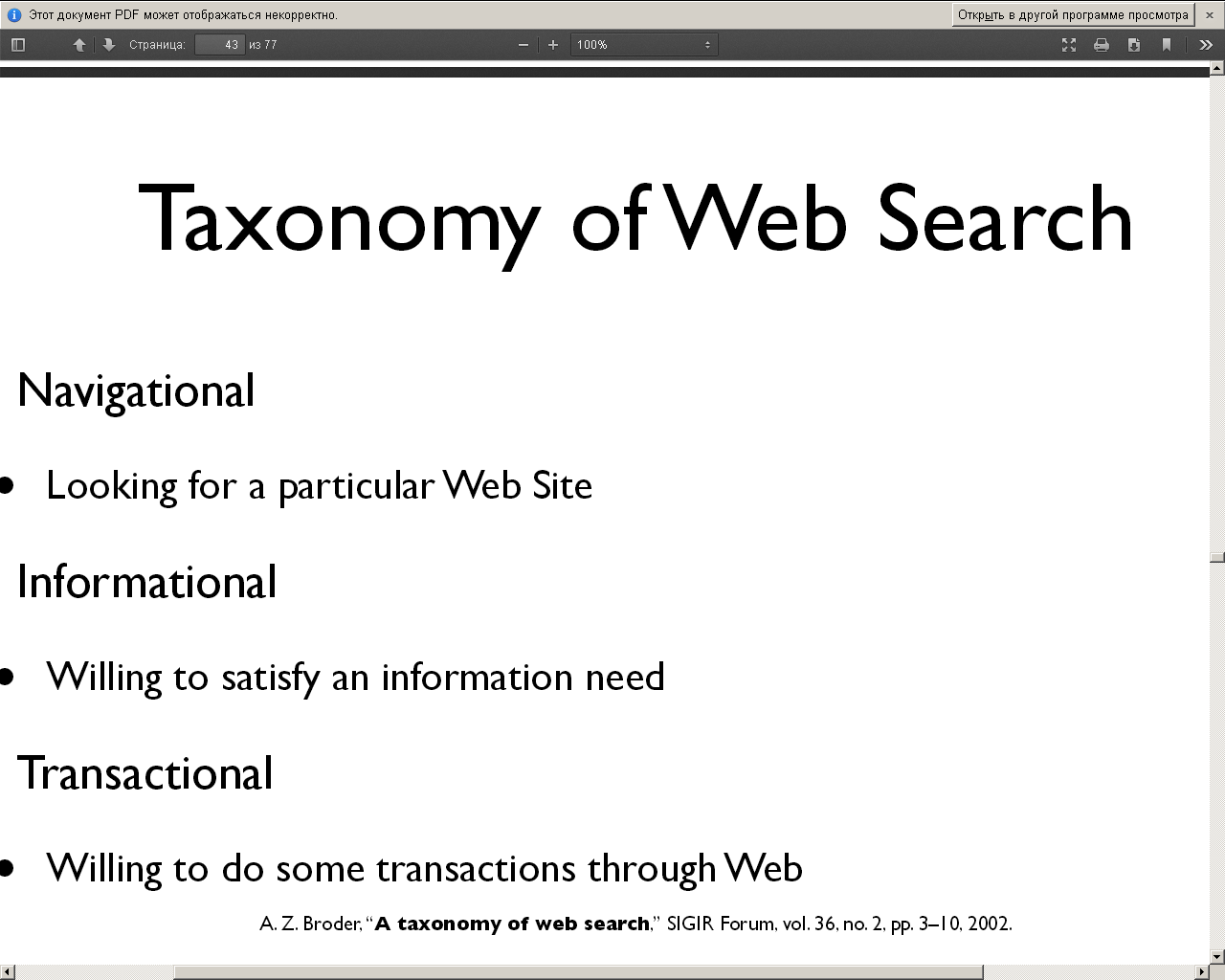
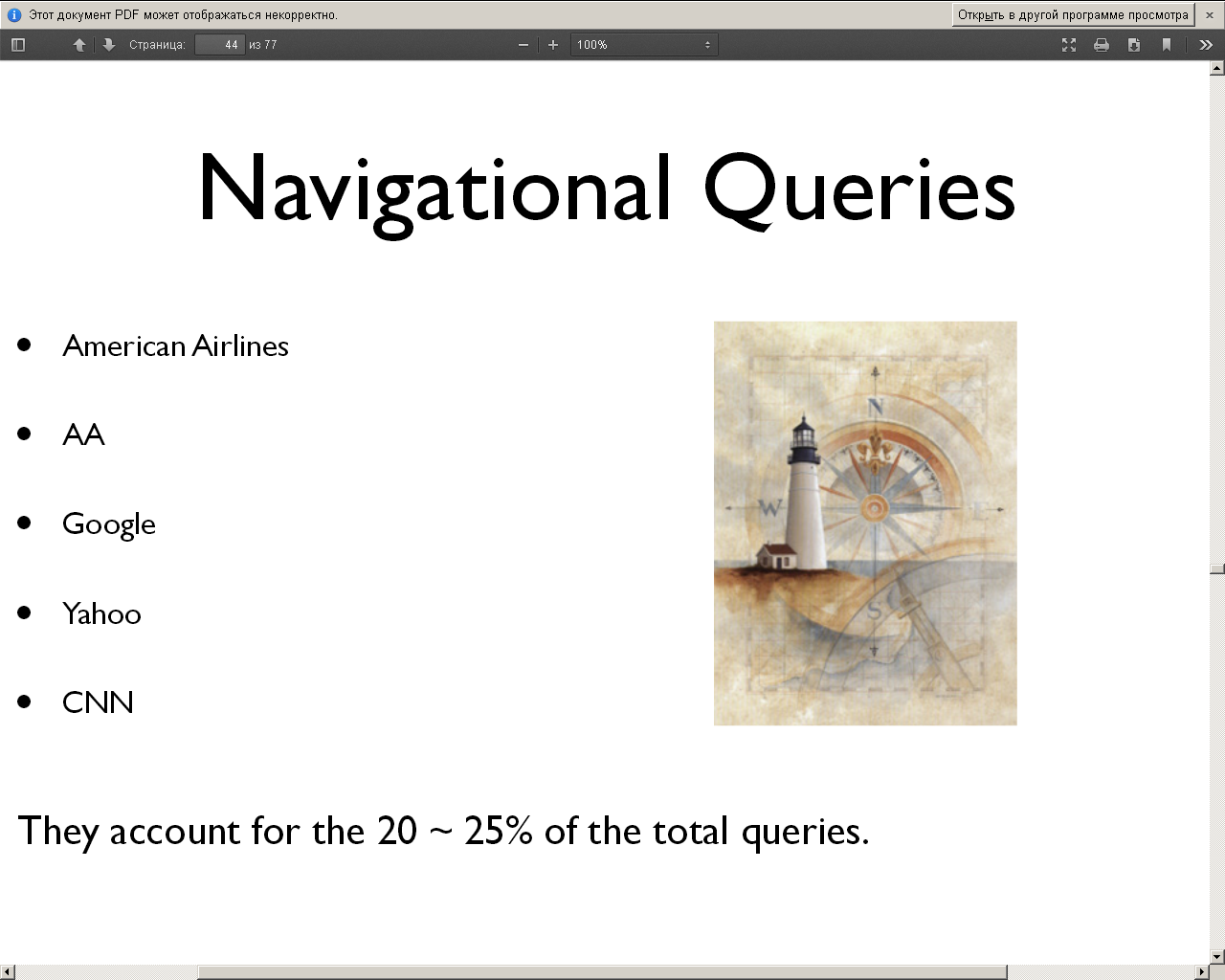
22.

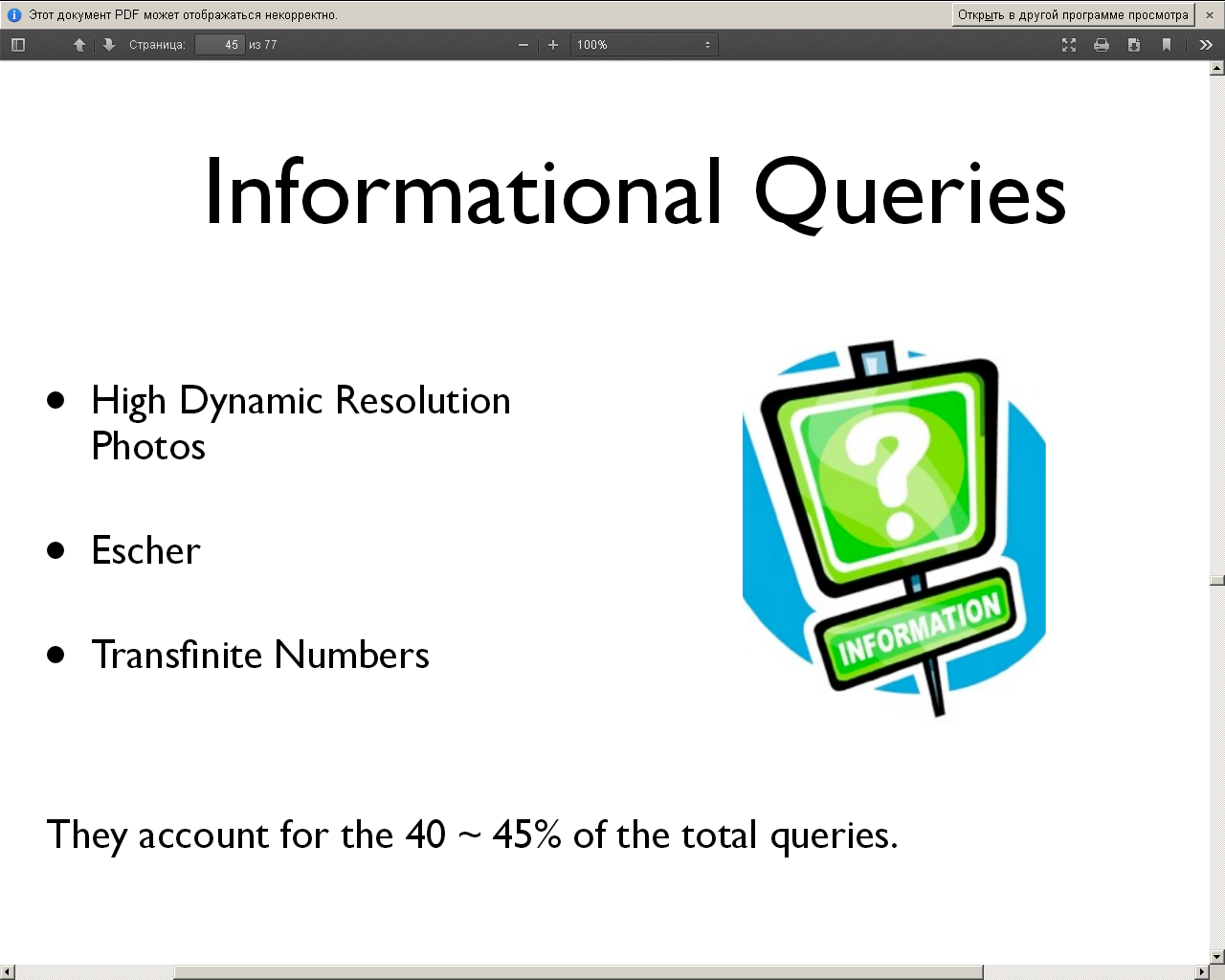
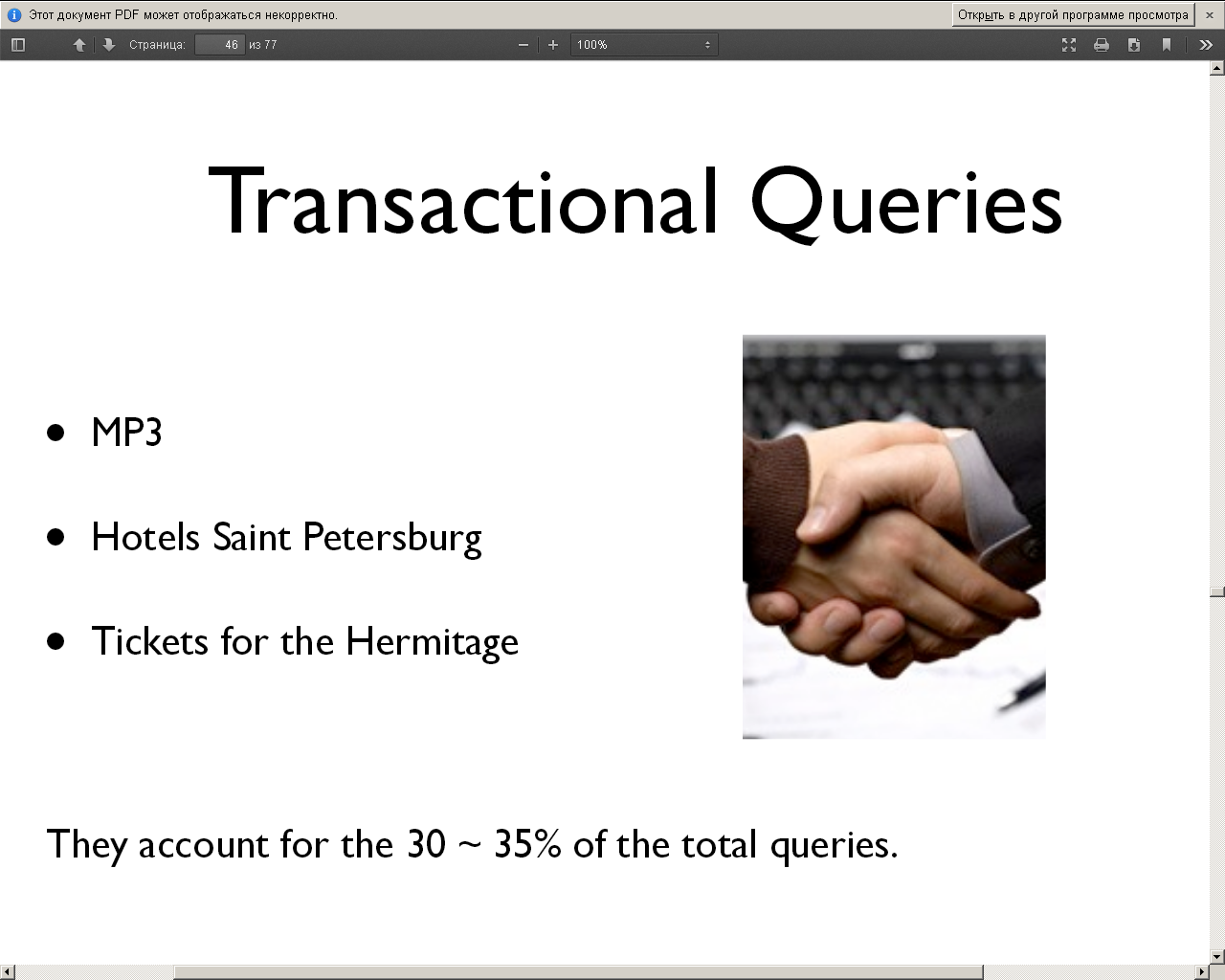
23.

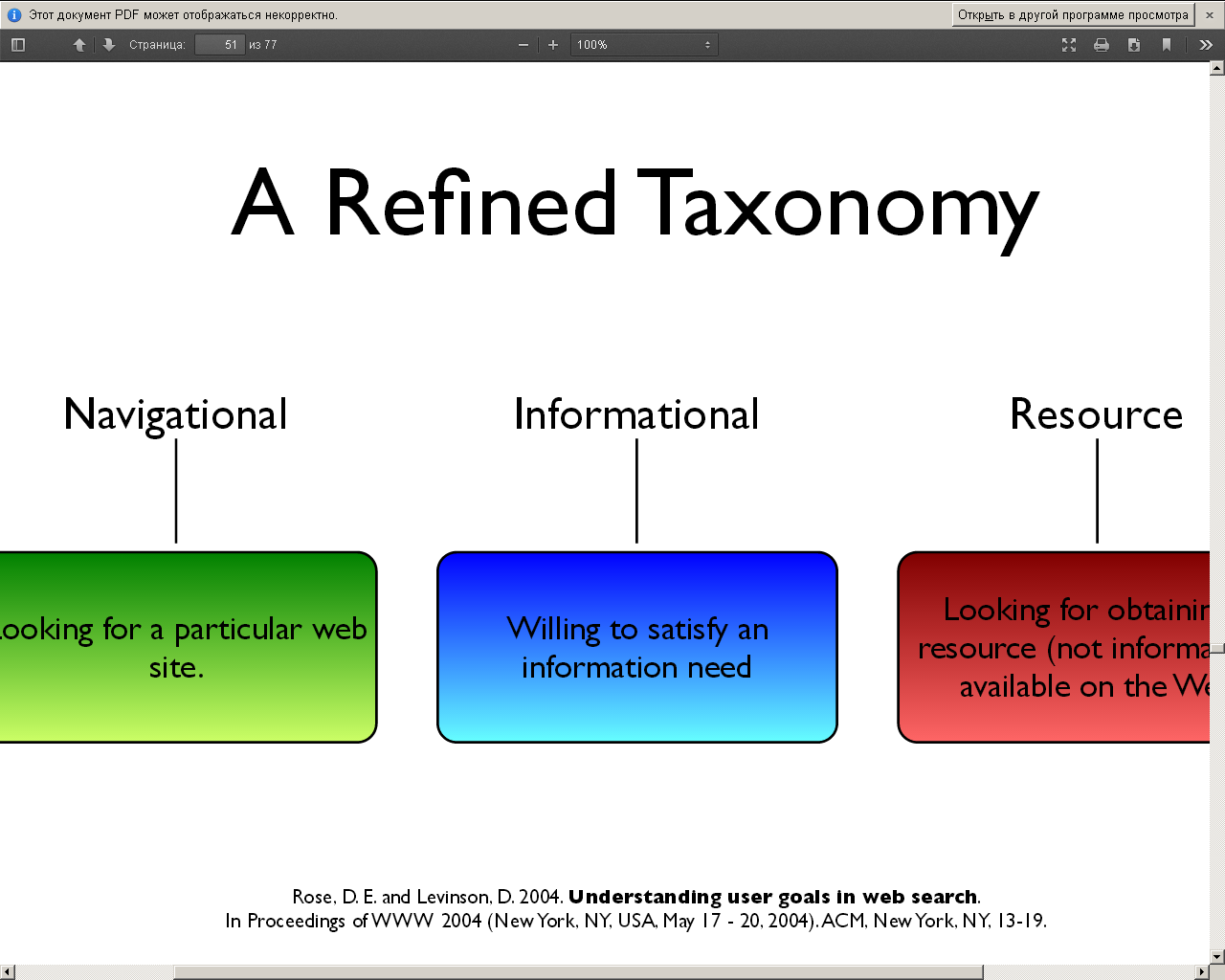
24.

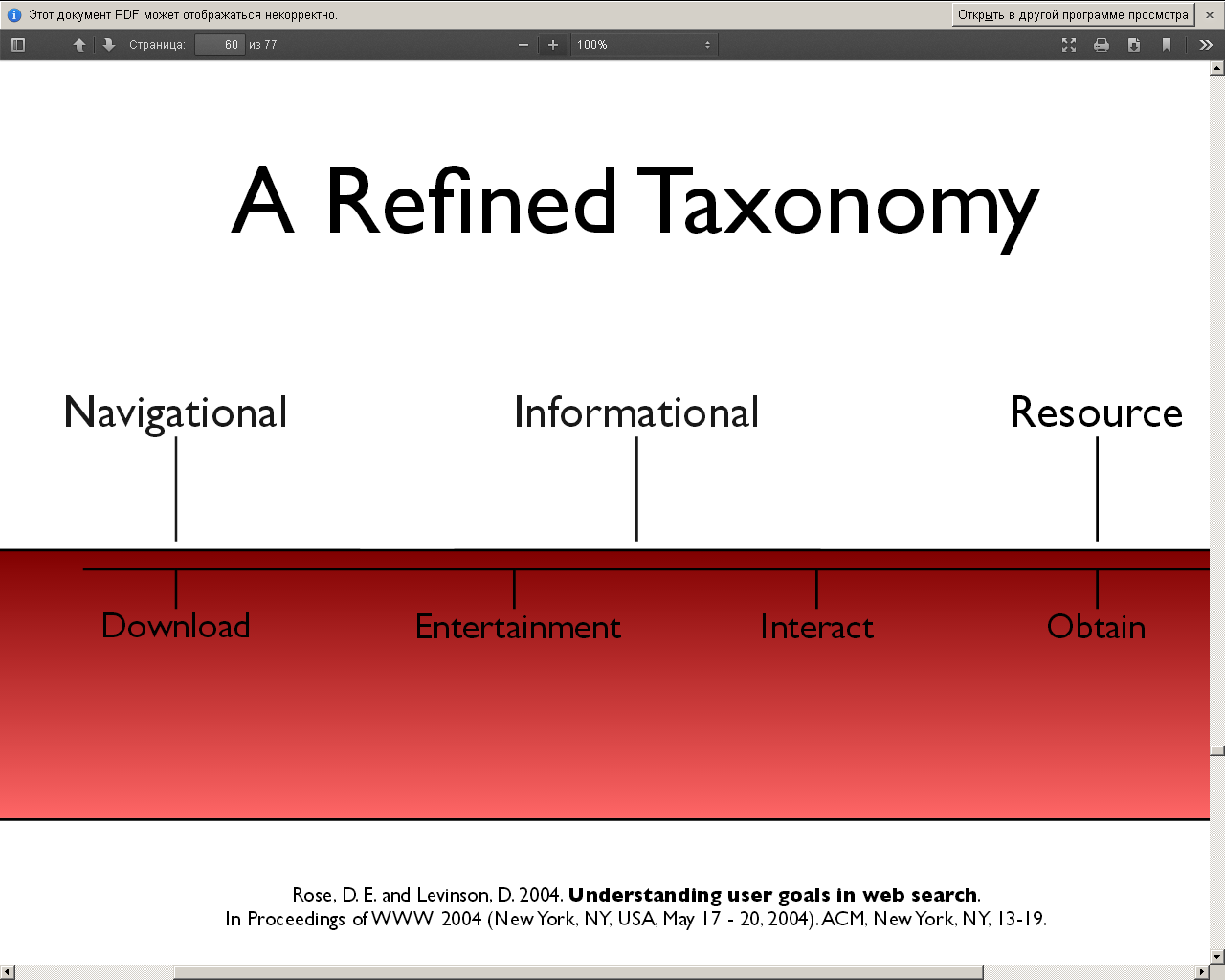
25.

25. Классификация запросов по цели. Зачем нужна. Особенности обработки разных типов запросов

 11



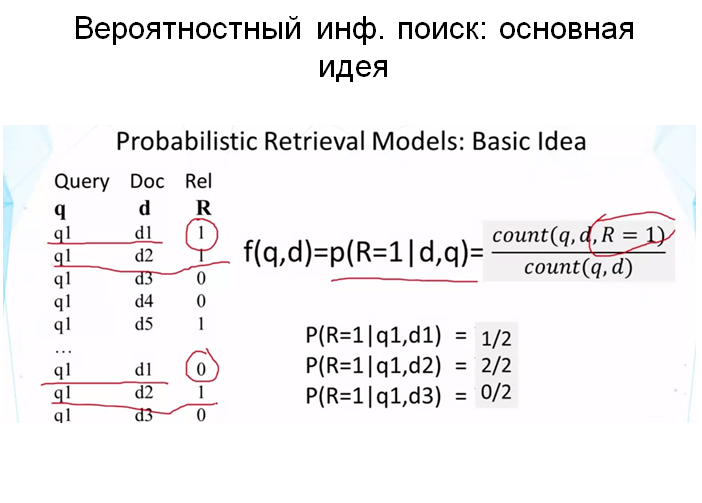
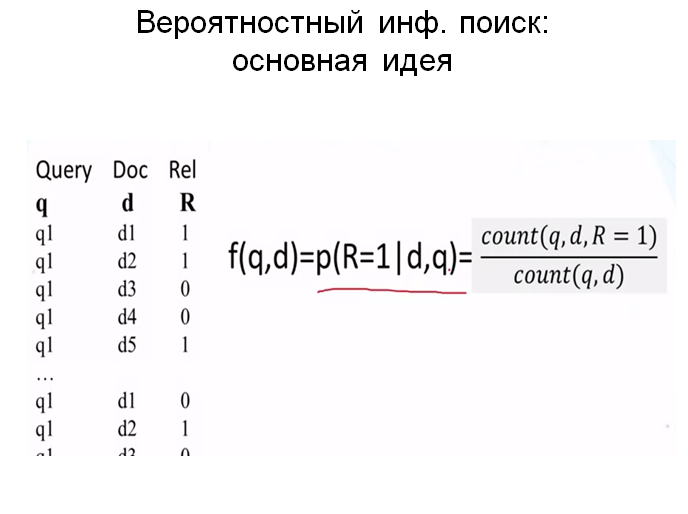


26. Вероятностная модель информационного поиска: основная идея, различие с векторной моделью

Векторная модель – выглядит как эвристика

Вероятностные модели

* + Попытка найти математическое обоснование
  + Классическая модель BIM (Binary Independence Model)
  + Модель BM25
  + Языковая модель информационного поиска

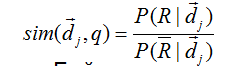


Определения

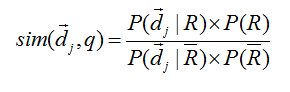
* + Все веса слов - бинарные, т.е. *wi,j* ∈ {0,1}
  + Пусть *R* – множество документов, про которые известно, что они релевантны запросу *q*
  + Пусть - оставшиеся документы
  + - это вероятность, что документ *dj* релевантен запросу *q*

- вероятность, что документ *dj* нерелевантен запросу *q*

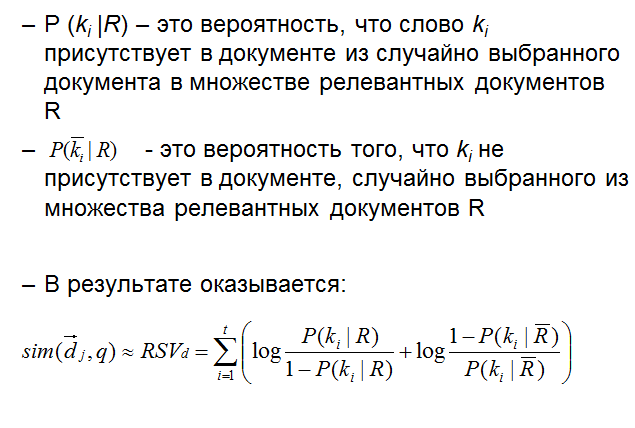
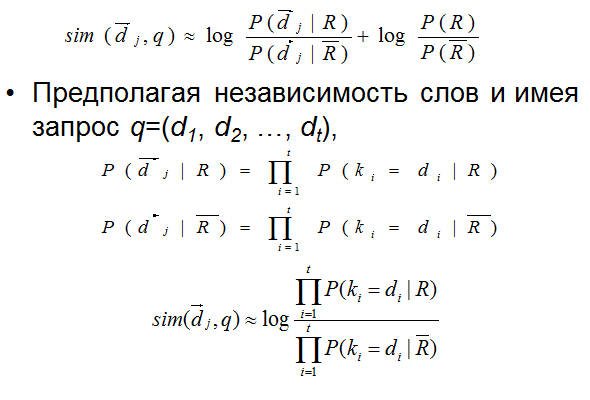
* *Сходство sim(dj,q) документа dj с запросом q определяется как отношение*



* Используя правило Байеса



* + *P(R)* – вероятность случайно выбрать релевантный документ в коллекции
  + - вероятность случайно выбрать документ *dj* из множества *R* релевантных документов



Оценки вероятности на практике

* Полные множества релевантных и нерелевантных документов неизвестны
  + Нужно оценивать
* Для нерелевантных документов (второе слагаемое)
* Если нерелевантные документы аппроксимировать целой коллекцией, то
* *ri* (вероятность встречаемость слова в нерелевантных документах для запроса)= df*/N*
  + log (1– *ri*)/*ri* = log (N– df)/df *≈* log N/df = IDF!

Статистика релевантных документов может быть оценена различным образом:

* + Можно использовать статистику слов в известных релевантных документах - это основа для вероятностных подходов к relevance feedback
  + Установить как константу. Предположим, что вероятность нахождения слова запроса в релевантном документе Pr(*ki* |*R*)= 0.5
    - Тогда первое выражение сокращается
    - Слабая оценка, но не противоречит предположениям
    - Получается, что ранжирование документов получено просто суммированием весов idf
    - Для коротких документов (заголовков или абстрактов) работает неплохо

Различие между векторными моделями и вероятностными не очень велико

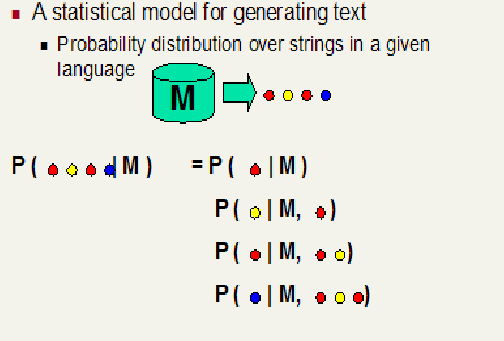
* + - В обоих случаях поисковая система строится похожим образом
    - Различия: в вероятностном информационном поиске сходство между запросов и документом считается не косинусной мерой и tf-idf в векторном пространстве, а несколько другой формулой, мотивированной теорией вероятности

27. Языковые статистические модели. Сглаживание. Где применяются

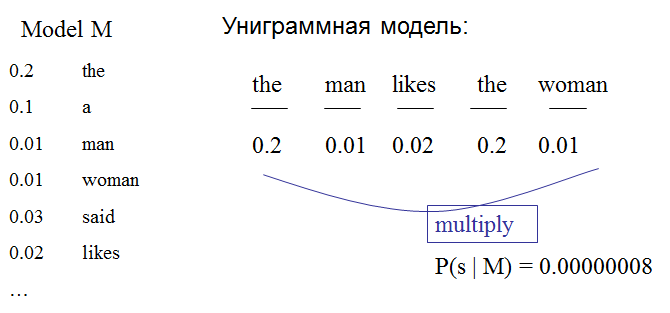
Статистические модели: определение вероятности предложений, последовательностей слов

* Как вероятна каждая последовательность?
  + P (w1, w2, w3,.. wn)
  + P(w5| w1, w2, w3, w4)
* Языковая модель – математическая модель, которая вычисляется вероятность последовательности слов или условную вероятность следования слова в контексте

Статистическая модель порождения текста определяет вероятности строк для данного языка

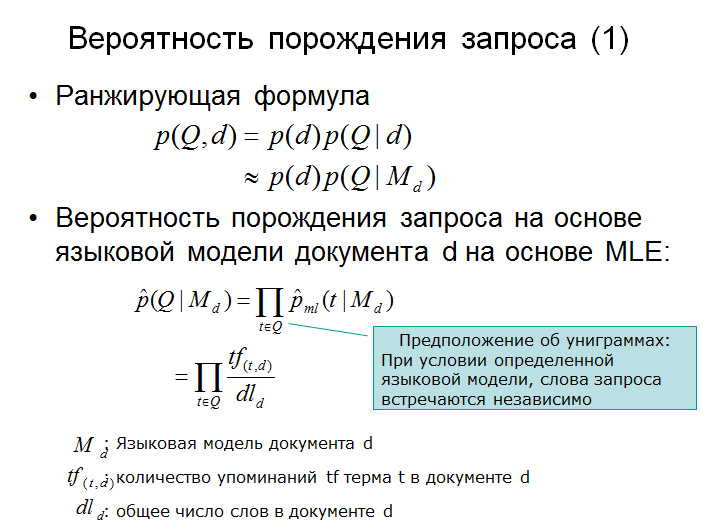


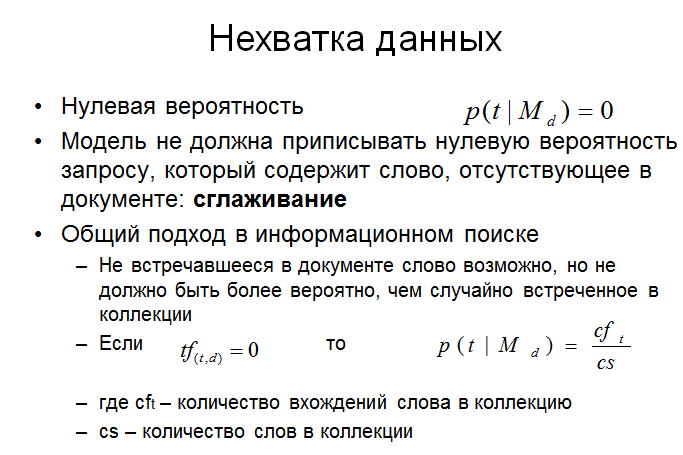
Статистические языковые модели Моделируют вероятность порождения строк в языке



Использование языковых моделей в информационном поиске

* Трактуем каждый запрос как случайный процесс
* Подход:
  + Насчитывает языковые модели для каждого документа
  + Выводим вероятность порождения запроса на основе модели каждого документа
  + Ранжируем документы в соответствии с этими вероятностями
  + Обычно используются униграммы

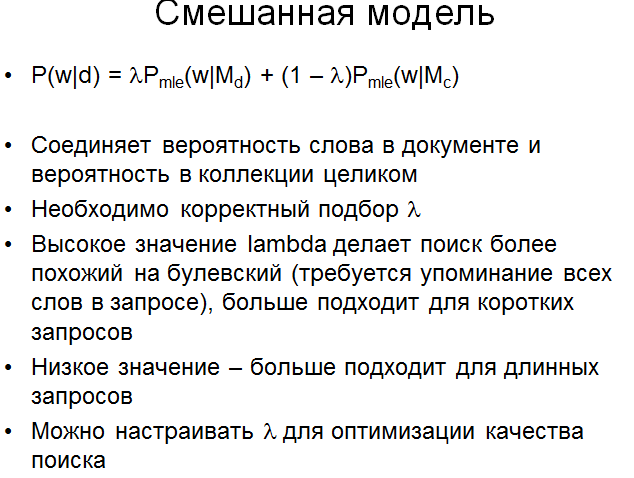


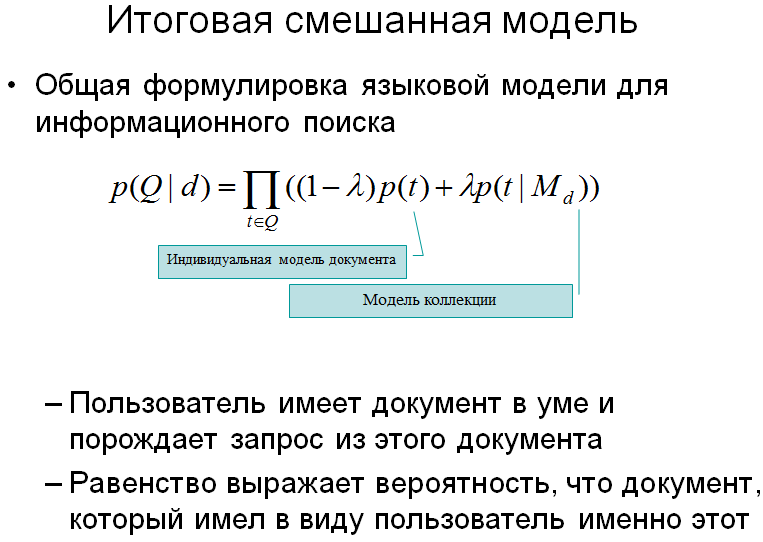


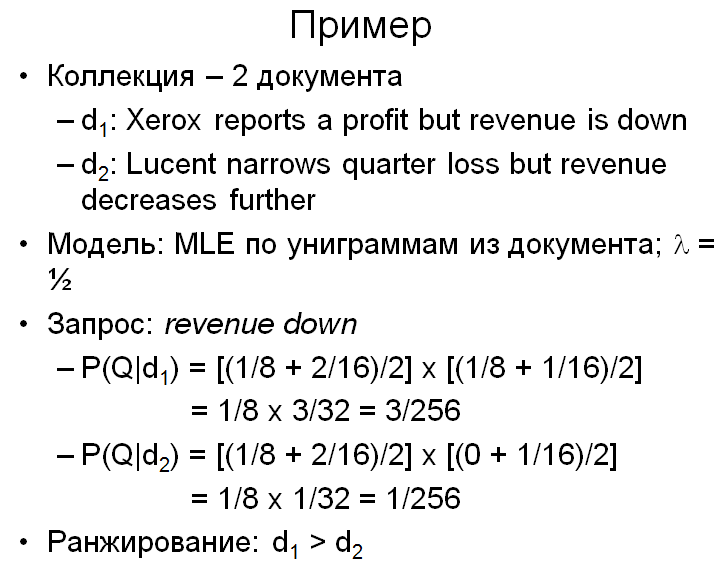
Сглаживание

* Необходимо уйти от нулевых вероятностей => техники сглаживания
* Уже упоминались при обсуждении байесовской классификации
* Методы
  + Ранее использовалось: добавить 1, ½ или ε к частотам упоминания
  + Здесь: смешивание статистики от документа и коллекции

28. Языковая модель информационного поиска







29. Методы приблизительного вычисления сходства документов в реальных поисковых системах

Неточная векторная модель: общий подход

* Найти А кандидатов *K < |A| << N*
  + А не обязательно содержит все К лучших документов, но содержит много документов из К
* Выдать К лучших документов из А
* Методы
  + Рассмотреть только слов с высокими idf
  + Рассмотреть только документы с большим количеством терминов запроса

Термы с высоким idf

* Запрос: *Над пропастью во ржи*
* Рассматривать только веса: *пропасть* и *рожь*
* Интуиция : над и во мало вносят в вес документа и не сильно изменят ранжирование
* Преимущества
  + Частотные слова содержатся в большом количестве документов
  + Резкое сокращение просмотра документов

Документы, содержащие много термов запроса

* Любой документ, содержащий хотя бы один терм запроса – кандидат в выдачу
* Для многословных запросов – считать только документы, в которых содержится несколько слова
  + Например, 3 из 4
  + Т.н. мягкая конъюнкция – можно часто встретить в интернет поиске

Champion lists

* Заранее для каждого терма t вычисляются r документов с максимальным весом t
  + Топ документов для t
  + r – выбирается во время индексирования
  + r не обязательно то же самое для всех термов
* Во время запроса, считаются только веса документов, входящих хотя бы в один топ-лист одного их термов запроса
  + Выбирается K лучших документов (наиболее похожих на запрос) из документов топ-листов

Учет качества документа

* Авторитетность – качество документа, не зависимое от запроса
* Примеры факторов:
  + Страницы Википедии
  + Статьи из заданного списка СМИ
  + Научные статьи с большим количеством цитирования
  + Сайты с большим количеством входящих ссылок

Моделирование авторитетности

* Присвоим каждому документу не зависимый от запроса вес качества [0,1]
  + Обозначим *g(d)*
  + Например, это количество цитат, нормализованное в шкале [0,1]
* Тогда можно считать net-score:
  + *net-score=α\*g(d)+ (1- α)\* cosine (q, d)*
* Теперь выбираем лучшие K документов по *net-score*

Лучшие К документов по   
net-score: идеи

1) Упорядочим все документы, соответствующие слову по мере снижения g(d)

* + Топовые документы в первых позициях

2) Храним список лучших документов для слова, по net-score

Ищем лучшие K-результатов только из этого списка

3) Для каждого документа храним два списка документов high и low

* При проходе по документам проходим сначала по high спискам
  + Если получили K документов, то ОК
  + Иначе начинаем обрабатывать документы из low списков
* При проходе по документам проходим сначала по high спискам
  + Получаются как бы списки разных уровней

30. Обработка фразовых запросов и запросов с указанием близости слов в поисковых системах

Фразовые запросы

* Предположим мы хотим найти ответ на запрос «Stanford University» (в кавычках)
* Тогда предложение “I went to university of Stanford” не подходит
  + Фразовые запросы
  + Много запросов пользователей внутри себя содержат фразовые запросы
* Не достаточно хранить только матрицу (индекс): терм-документ

Подходы к обработке фраз:  
 биграммный индекс

* Нарезаем тексты на пары слов, строим на парах индекс: *I went, went to, to university, university of, of stanford*.
* Длинные запросы также нарезаем на пары слов: stanford university palo alto=>
* stanford university AND university palo AND palo alto
* !Но могу встретиться документы, которые содержат пары слов, но не содержат фразу
* Проблемы:
  + сверхбольшой индекс
  + Проблемы со случайным вхождением слов внутрь биграммы
  + не является стандартным решением

Обработка запросов с операторами близости

* *Москва /3 университет*
* Только позиционный индекс может использоваться для такой обработки, биграммный не может
* Позиционный индекс сейчас стандартно используется для обработки запросов на близость и фразовых запросов
* Возможно комбинирование биграммных и позиционных индексов



Близость слов запроса

* Запросы на естественном языке: набор терм, набиваемых в поисковую строку
* Пользователь предпочитает, чтобы термины запроса встречались недалеко друг от друга
* Пусть *w* – минимальное окно, содержащее все слова запроса, например,
* Для запроса *увольнение директора* минимальное окно в документе *Директора ждало неожиданное увольнение* – 4 слова
* Как учесть в скоринговой функции?

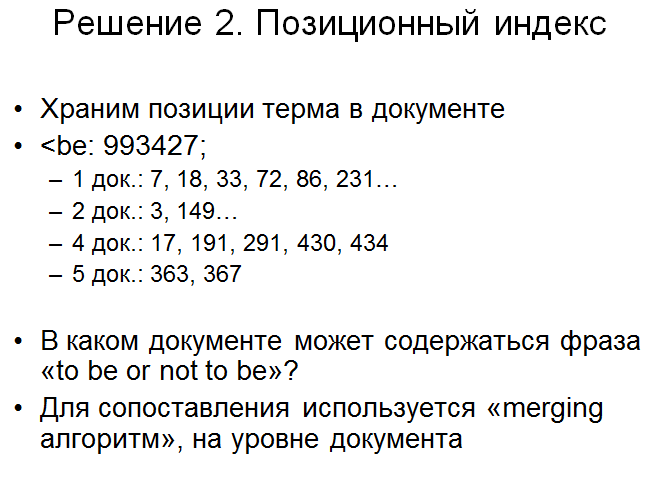
Разборщики запросов (колдунщик)

* Текстовый запрос пользователя может исполниться посредством нескольких запросов к системе, например, запрос: *повышение оплаты труда*
  + Исполняем запрос как фразовый запрос
  + Если <K документов содержат фразу повышение оплаты труда, то исполняются два фразовых запроса *повышение оплаты* и *оплата труда*
  + Если все еще меньше К документов, то запрос на векторное пространство *повышение оплаты труда*
  + Эти операции исполняет разборщик запросов
  + Скоринговая функция должна учитывать разные факторы, включая минимальное окно, в котором содержится запрос

Агрегирование весов:

* Много разных факторов
  + Скалярное произведение
  + Близость слов запроса
  + Ссылки
  + Соответствие тематики запроса и документа

31. Позиционный индекс в поисковой системе. Зачем нужен, как обрабатывается

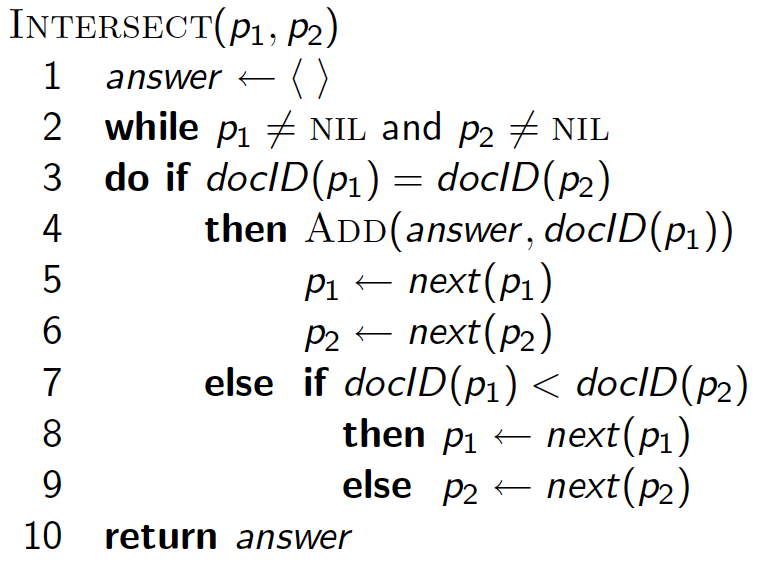


* Извлекаем инвертированный индекс для слов *to, be, or, not*
* ***to****:*
* *2*:1,17,74,222,551; ***4*:8,16,190,429,433**; *7*:13,23,191; ...
* ***be****:*
* *1*:17,19; ***4*:17,191,291,430,434**; *5*:14,19,101; ...

Обработка запросов с операторами близости

* *Москва /3 университет*
* Только позиционный индекс может использоваться для такой обработки, биграммный не может
* Позиционный индекс сейчас стандартно используется для обработки запросов на близость и фразовых запросов
* Возможно комбинирование биграммных и позиционных индексов

Merge algorithm



**Задачи**

**Кластеризация**

* Начинает с рассмотрения документов как отдельных кластеров
  + итеративно объединяет ближайшую пару кластеров, до тех пор пока не останется один кластер.
* История объединения и образует бинарное дерево или иерархию

Ближайшая пара кластеров

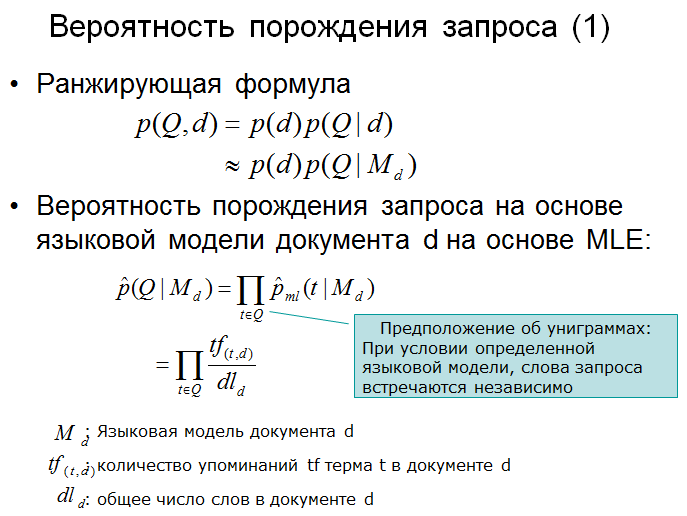
* Много способов определения, что такое наиболее сходная пара кластеров
* Single-link
  + Сходство по наиболее похожим документам (single-link)
* Complete-link
  + Сходство по наиболее непохожим документам
* Центроид
  + Сходство по наиболее похожим центроидам
* Average-link
  + Средний косинус между парами элементов двух кластеров

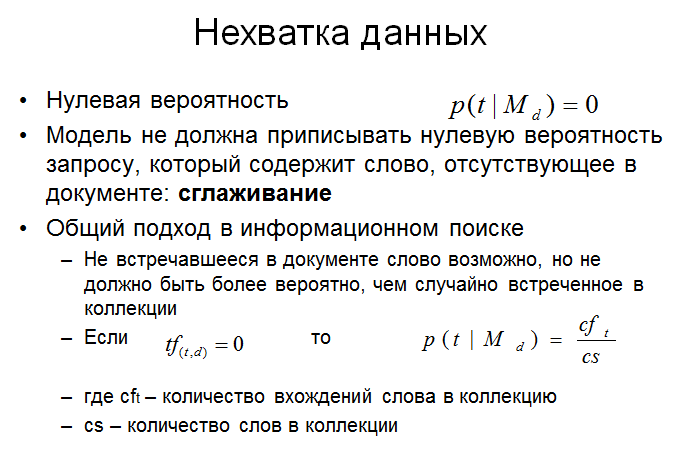
**Аггломеративная кластеризация: Single Link**

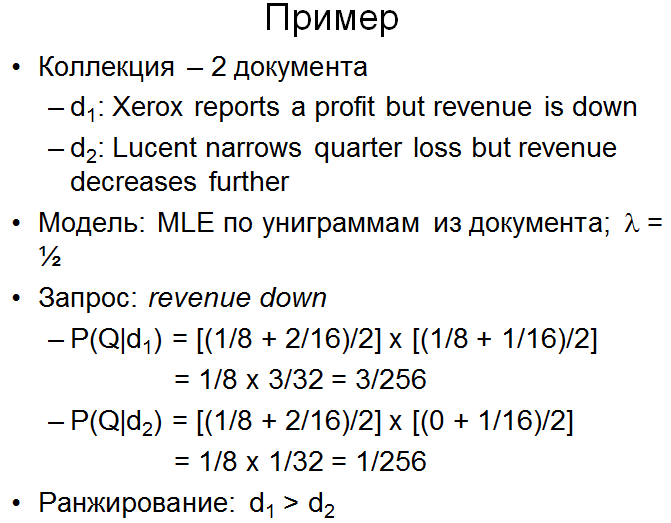
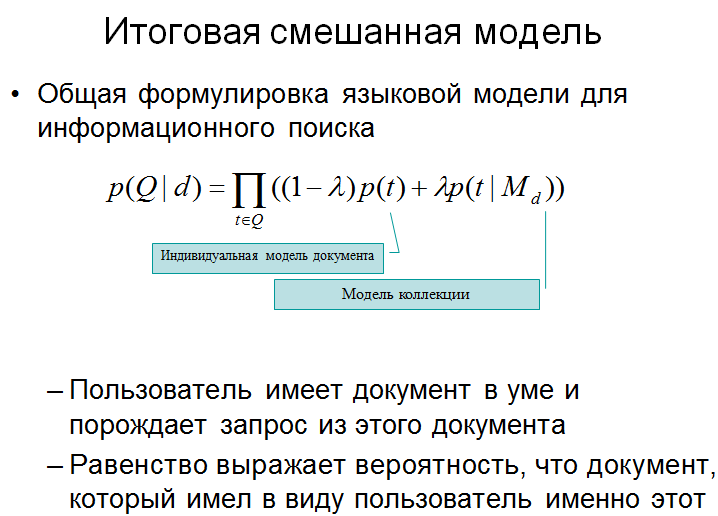
* Использует максимальное сходство пар
* Может породить длинные и тонкие кластеры - цепочки.
* После склеивания *ci* и *cj*, сходство результирующего кластера к другому кластеру, *ck*, выбирается как максимальное из расстояний между ci и ck и cj и ck.

**Кластеризация по всем связям (complete link)**

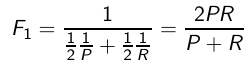
* Использует наименее сходные пары
* Создает более «плотные», сферические кластеры.
* После склеивания *ci* и *cj*, сходство результирующего кластера с другим кластером, *ck*, выбирается как минимальное из расстояний между ci и ck и cj и ck.

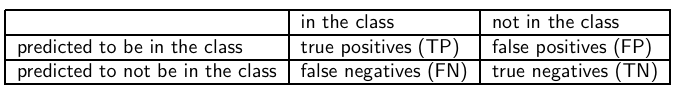
5. Вычисление близости документа к запросу по языковой модели информационного поиска 





2





*P* = *TP* / ( *TP* + *FP*)

*R* = *TP* / ( *TP* + *FN*)

* 1. Посчитали меру оценки (*F*1) for one class.
  2. Как агрегировать оценки F1 для многих классов.
  3. Macroaveraging - макроусреднение
     1. Посчитать *F*1 для каждого из *C* классов
     2. Среднее арифметическое для этих *C* чисел
  4. Microaveraging - микроусреднение
     1. Посчитать TP, FP, FN для каждого из *C* classes
     2. Суммировать эти С чисел для каждого показателя
     3. Посчитать *F*1 для суммированных TP, FP, FN

7

Вероятностная модель BIM была изначально создана для поиска по записям в коротких каталогах сопоставимой длины – и работала прилично в этих условиях

Для современного полнотекстового поиска, модель должна учитывать частоту термина в документе и длину документа

BestMatch25 (BM25 или Okapi), развитие модели BIM, учитывает эти величины

С 1994 до наших дней, модель BM25 – это одна из наиболее распространенных и устойчивых моделей информационного поиска

Простейшая форма веса для документа d – это просто суммирование idf слов запроса, которые присутствуют в этом документе: 1130.png

Это формула «исправляется» учетом частоты слова в документа и длины документа:

11302.png

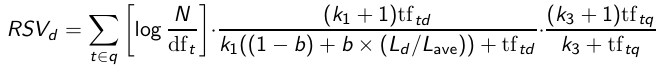
tf *td* : частота слова в документе d

*Ld* (Lave): длина документа d (средняя длина документа в коллекции)

*k1*: параметр, контролирующий учет частоты слова

*b*: параметр, контролирующий учет длины документа

Если запрос длинный, то можно учитывать похожее взвешивание для слов запроса



tf *tq*: частота слова в запросе q

*k3*: параметр, контролирующий частоту термина в запросе

Нет нормализации запроса по длине (поскольку поиск делается для фиксированного запроса)

Параметры нужно настраивать на коллекции

Если оптимизация не выполнялась, то в экспериментах получено, что величины *k1* и *k3* д