Введение

Решается задача ранжирования текстов по запросам пользователей. Решения данной задачи описаны в [4, 5, 6]. При построении учитывались особенности запросов пользователей, однако эти модели сталкивались с проблемой переобучения.

Модели высокого качества также были найдены с помощью алгоритма полного перебора. В работе [7] такие модели рассматриваются как суперпозиции математических примитивов от основных характеристик текста — частоты слова в документе — (tf) и числа документов, в которых встречается слово — (idf). Для моделей вводилась сложность модели — число элементов грамматики, используемых для их описания. Накладывались структурные и целевые ограничения. Лучшие модели, описанные в [7], превосходят по качеству на коллекциях TREC модели из работ [4, 5, 6]. Однако более детальное исследование пространства структурно сложных суперпозиций — нетривиальная задача.

Одним из подходов к поиску оптимальной модели является генетический алгоритм. Он основан на идее итеративного отбора моделей, их скрещивания и мутаций. Первые попытки его использования были произведены в статьях [9, 10]. Производились поиски наилучших параметров генетического алгоритма. Как показано в [11, 12], оптимальный выбор операции кроссовера может существенно улучшить порождаемые модели.

Основным преимуществом генетического алгоритма является гибкость порождаемых им моделей, что позволило в [13, 14] перейти к представлению ранжирующей модели ее деревом синтаксического разбора. Однако генетический алгоритм, описанный в [13, 14], подвержен стагнации. После 30-40 итераций мутаций и кроссовера сложность порождаемых функций значительно возрастает и изменения в популяции становятся незначительными.

Улучшения этого метода описаны в статье [8], где благодаря использованию регуляризации функционала качества моделей, удается добиться улучшения разнообразия порождаемых функций, что ведет к повышению качества итоговой модели.

В работе [15] рассматривается другой подход к аналитическому программированию. Предлагается использовать принципы глубокого обучения, разбивая задачу по уровням абстракции. В качестве промежуточного уровня предложено использовать матрицу вероятностей переходов в дереве разбора суперпозиции, полученную обучением нейронной сети. Далее, итоговая модель строится итеративным жадным алгоритмом. (TODO)

Предлагается альтернативная реализация генетического алгоритма со следующими изменениями <тут наши предложения по генетике>.

В качестве основной новации рассматривается развитие идеи предсказания промежуточной мета-модели <тут наши предложения>

Работа построена следующим образом. <Описание структуры>

Базовая постановка задачи

Дана коллекция текстовых документов $\mathbf{C} = \{d_i\}$ и пользовательских запросов $\mathbf{Q}$, каждый из которых представляет из себя множество слов $q = \{w_i\}$. Дана функция $r(d, q) \to \{0, 1\}$, определенная экспертами, и показывающая, является ли данный документ d релевантным для запроса q (1 — является).

Рассмотрим две характеристики пары документ-слово: $(d, w, \mathbf{C}) \to (\text{tf}, \text{idf})$. Определенных следующим образом:

$$\text{idf}(w, \mathbf{C}) = \frac{\text{count}(w, \mathbf{C})}{|\mathbf{C}|}$$

$$\text{tf}(w, d, \mathbf{C}) = freq(w, d) \cdot \log(1 + \frac{\text{size}_{\text{avg}}}{\text{size}(d)})$$

где $\text{count}(w, \mathbf{C})$ — количество документов $d \in \mathbf{C}$ содержащих слово w, $\text{freq}(w, d)$ — частота вхождения слова w в документе d, $\text{size}(d)$ — количество слов в d, а size_{avg} — среднее количество слов в документах из коллекции C.

Положим f — суперпозиция математических функций от аргументов tf и idf. Назовем моделью — дерево синтаксического разбора данной суперпозиции и рассмотрим множество всех таких деревьев $\mathcal{T}$.

Будем аппроксимировать функцию $r(d, q)$, как функцию $f(d, q) = \sum_{w \in d} f'(\text{tf}, \text{idf})$, где $f' \in \mathcal{T}$.

Качеством аппроксимационной функции будем считать MAP (mean average precision).

$$\text{MAP}(f, \mathbf{C}, \mathbf{Q}) = \frac{1}{|\mathbf{Q}|} \cdot \sum_{q \in Q} \text{AvgP}(f, q, \mathbf{C})$$

$$\text{AvgP}(f, q, \mathbf{C}) = \frac{\sum_{i=0}^{|C_q|} \text{PrefSum}(r(d_{(i)}, q), k) \cdot r(d_{(i)}, q)}{\sum_{d \in C_q} r(d)}$$

Где C_q — множество документов коллекции, размеченных для запроса q, $d_{(i)}$ — i-ый документ из C_q в ряду, упорядоченному по убыванию значения $f(d_{(i)}, q)$, $\text{PrefSum}(r(d_{(i)}, q))$ — сумма первых k элементов ряда $\{r(d_{(i)}, q)\}$.

Нашей задачей является нахождение ранжирующей функции

$$f^* = \arg\max_{f \in \mathcal{T}} \big(\text{MAP}(f, \mathbf{C}, \mathbf{Q}) - P(f)\big)$$

где $P(f)$ — штрафная функция, ограничивающая структурную сложность суперпозиции f.

Постановка задачи на кластерах документов

Определим $\text{tf} - \text{idf}$ для всей коллекции документов способом аналогичным рассмотренному выше. Фактически рассмотрим отображение $V : \mathbf{C} \to \mathbb{R}^n$. Где каждому документу сопоставляется вектор $\text{tf} - \text{idf}$ представления всех слов в нем.

Кластеризуем документы, используя их представление в пространстве $\mathbb{R}^n$. Расстояние между документами считаем при помощи стандартной эвклидовой метрики.

Получаем множество кластеров $D = \{d_i : d_i = \{c_j \in C\}\}$, $|D| = m$. Построим для каждого кластера семейство ранжирующих функций $F^*_{d_i} = \{f_i^1, \cdots, f_i^n\}$, используя генетический алгоритм. В каждом семействе выделим наилучшую ранжирующую функцию $f_i^* \in F_{d_i}$.

Определим ранжирующую функцию на кластерах:

$$f^* = \arg\max_{W \in \mathbb{R}^m} \big(\text{MAP}(\sum_i W_i * f_i^*, \mathbf{C}, \mathbf{Q}) - \sum_i P(f_i^*)\big)$$

Оптимизацию весов W будем производить при помощи поиска по сетке.

Описание генетического алгоритма

При создании случайной суперпозиции генерируется случайное дерево малой глубины, при этом в каждом узле случайно выбирается одна из базовых операций. Так же выполняется случайное уменьшение глубины дерева. После данной процедуры получаются деревья из 15-30 узлов.

В качестве операции кроссовера для двух объектах популяции используется обмен случайных двух узлов в деревьях суперпозиций.

Алгоритм 1 Создание ранжирующией функции для коллекции документов

Вход: N_{epoch}, **C**
Выход: f^* - наилучшая модель в итоговой популяции
▷ Сгенерировать начальную случайную популяцию $\mathcal{T}_0$
повторять
▷ Выполнить кроссовер для случайной пары суперпозиций из $\mathcal{T}_\text{i}$
▷ Мутировать случайную суперпозицию из $\mathcal{T}_\text{i}$
▷ Отранжировать популяцию $\mathcal{T}_\text{i}$ согласно метрике MAP
▷ Учитывая регуляризацию по числу вершин
▷ Выбрать наилучшие суперпозиции, составить из них популяцию $\mathcal{T}_{\text{i}+1}$
▷ Увеличить число пройденных эпох
пока $epoch \neq N_{epoch}$

Используемые данные

Для обучения и тестирования модели используется коллекция текстовых документов TREC [16]. В частности коллекции Trec-5 – Trec-8. Для каждой коллекции представлены набор запросов, ранжирование документов, проведенное экспертами экспертами, и сами документы, на которых решается задача информационного поиска. Основой каждой коллекции является набор документов, предоставляемых NIST, являющимся спонсором конференции TREC. В каждой коллекции представлено в среднем 500 000 документов, 100 запросов и порядка 2000 ответов для каждого запроса. Номер каждой коллекции непосредственно связан с номером конференции, на которой рассматривалась данная коллекция.

Обработка данных

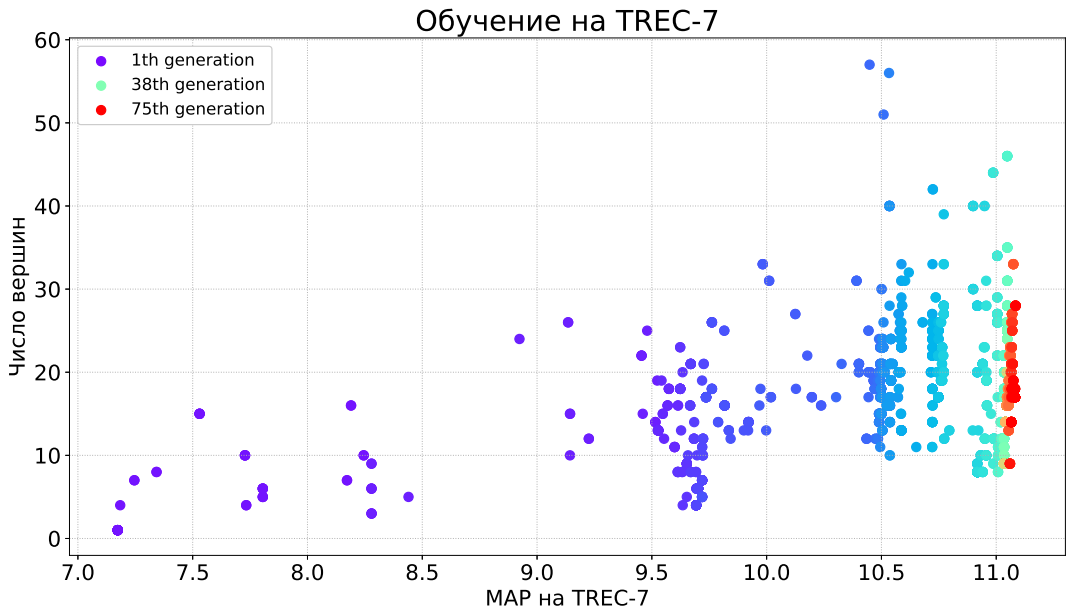
Обработка данных производилась средствами языка Python. Использовались стандартные библиотеки sklearn и nltk. Производилось удаление стоп-слов, соответствующих английскому языку, приведение их к начальной форме при помощи PorterStemmer'a. Для подсчета непосредственных значений tf использовался CountVectorizer, значение idf вычислялось непосредственно согласно формулам выше.

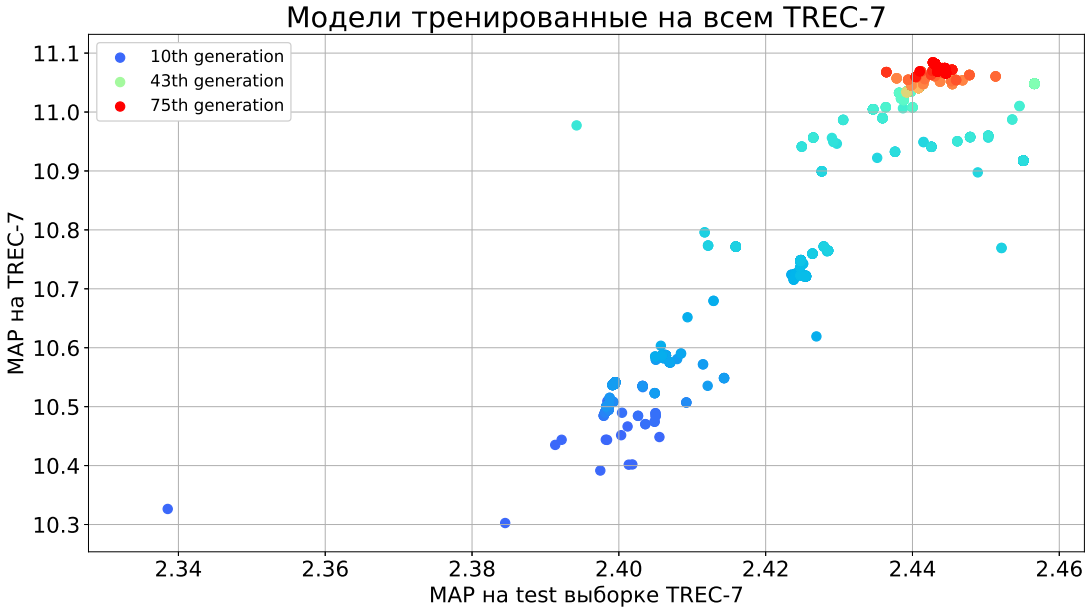
Алгоритм 2 Вычисление $\text{tf} - \text{idf}$ по корпусу документов

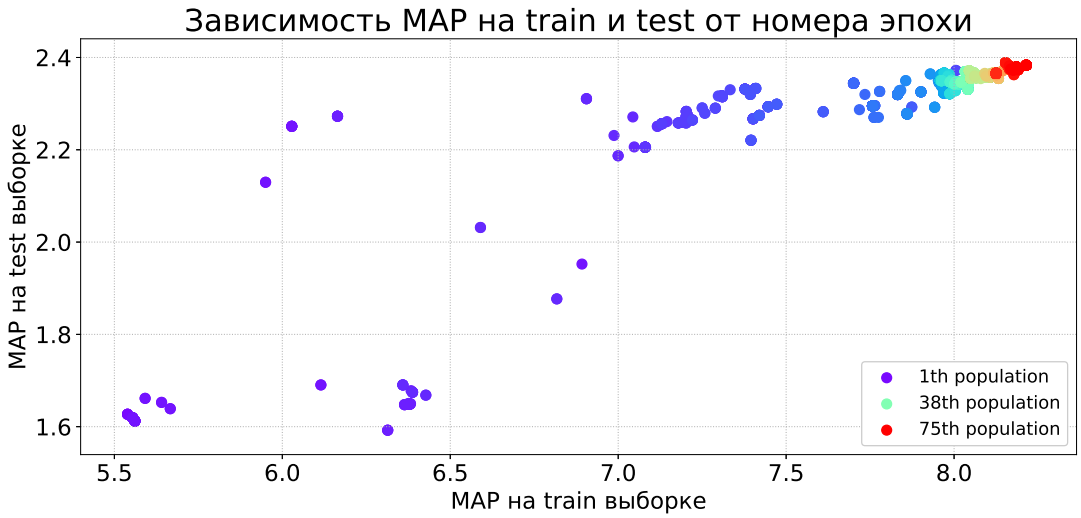
Вход: **C**
Выход: $\text{tf} - \text{idf}$
для каждого слова в коллекции документов **C**
▷ Удалить знаки препинания и служебные символы
▷ Разделить текст на слова
▷ Удалить стоп–слова
▷ Выполнить стемминг каждого слова
▷ Вычислить значения tf-idf по матрице слов для документов

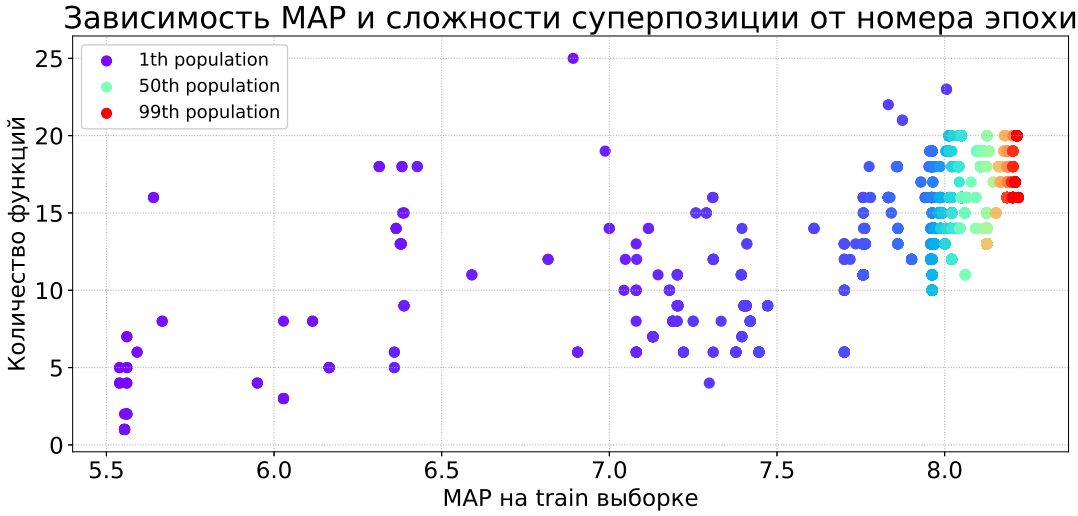
Базовый вычислительный эксперимент

Сначала был проведен эксперимент по исследованию качества популяции ранжирующих функций, генерируемых генетическим алгоритмом. В качестве основного был взят корпус документов TREC-7. Выборка была разделена на обучающую и валидационную в соотношении $80\% - 20\%$. Результаты обучения на данном корпусе приведены ниже.

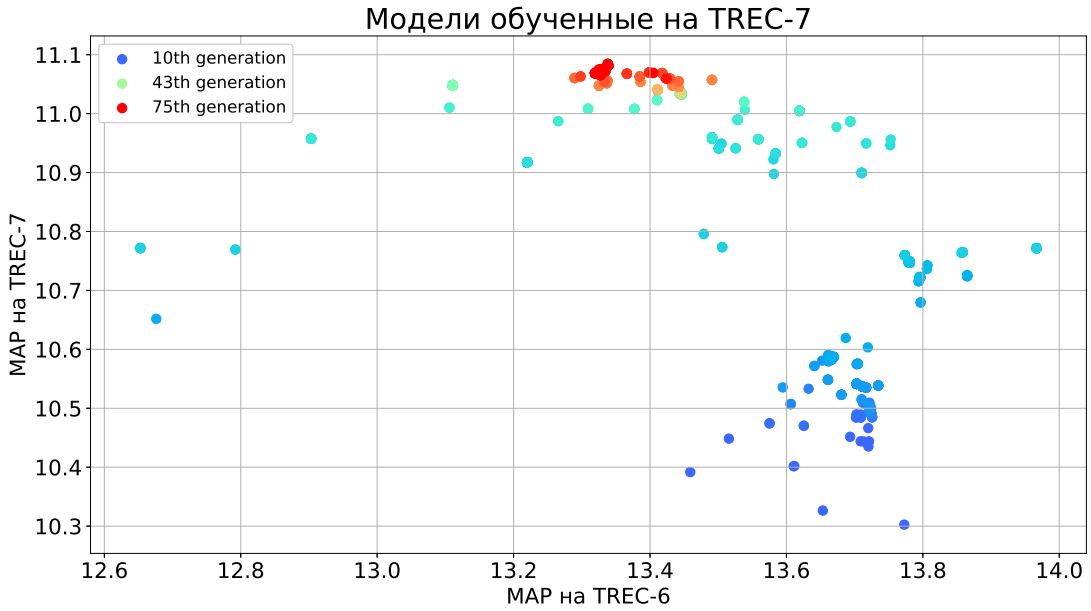

Обучение на TREC-7
1th generation
38th generation
75th generation
Число вершин
MAP на TREC-7
0
10
20
30
40
50
60
7.0
7.5
8.0
8.5
9.0
9.5
10.0
10.5
11.0


Модели тренированные на всем TREC-7
10th generation
43th generation
75th generation
MAP на TREC-7
MAP на test выборке TREC-7
10.3
10.4
10.5
10.6
10.7
10.8
10.9
11.0
11.1
2.34
2.36
2.38
2.40
2.42
2.44
2.46





Отсюда видно, что генетический алгоритм начинает сильно переобучаться, что подтверждается при тестировании моделей обученных на TREC-7 на корпусе TREC-6.



Так же приведена таблица сравнения с уже известными ранжирующими функциями сообщества. Наилучшие найденные функции:

Вычислительный эксперимент при кластеризации данных

Так как кластеризация данного объема данных достаточно трудоемкая задача, был использован простой алгоритм K-means [17]. Количество кластеров, на которые разбивалась выборка было выбрано <TODO>.

Результаты при кластеризации корпуса <TODO>.

Заключение

<TODO>

Литература

[1] *Porter M. F.* Readings in Information Retrieval // Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, Ch. An Algorithm for Suffix Stripping, Pp. 313–316.

[2] *Metzler, Donald and Croft, W. Bruce* A Markov Random Field Model for Term Dependencies // Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '05, ACM, New York, NY, USA, 2005, pp. 472–479.

[3] *Amati, Gianni and Van Rijsbergen, Cornelis Joost* Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness // ACM Trans. Inf. Syst. 20 (4) (2002) pp. 357–389

[4] *Salton, Gerard and McGill, Michael J.* Introduction to Modern Information Retrieval // McGraw-Hill, Inc., New York, NY, USA, 1986

[5] *Ponte, Jay M. and Croft, W. Bruce* A Language Modeling Approach to Information Retrieval // In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–281. ACM.

[6] *Clinchant, Stéphane and Gaussier, Eric* Information-based Models for Ad Hoc IR // In Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 234–241. ACM.

[7] *P. Goswami, S. Moura, E. Gaussier, M.-R. Amini, F. Maes* Exploring the space of ir functions // ECIR'14, 2014, pp. 372–384.

[8] *Kulunchakov A. S., Strijov V. V.* Generation of simple structured IR functions by genetic algorithm without stagnation // `http://strijov.com/papers/Kulunchakov2014RankingBySimpleFun.pdf`

[9] *Goldberg, David E.* Genetic Algorithms in Search, Optimization and Machine Learning // Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[10] *Koza, John R.* Genetic Programming: On the Programming of Computers by Means of Natural Selection // MIT Press, Cambridge, MA, USA, 1992.

[11] *Vrajitoru, Dana* Crossover Improvement for the Genetic Algorithm in Information Retrieval // Inf. Process. Manage. 34, 4 (July 1998), 405-415.

[12] *Gordon, M.* Probabilistic and Genetic Algorithms in Document Retrieval // Commun. ACM 31, 10 (October 1988), 1208-1218.

[13] *Fan, Weiguo and Gordon, Michael D. and Pathak, Praveen* Personalization of Search Engine Services for Effective Retrieval and Knowledge Management // In Proceedings of the twenty first international conference on Information systems (ICIS '00). Association for Information Systems, Atlanta, GA, USA, 20-34.

[14] *Fan, Weiguo and Gordon, Michael D. and Pathak, Praveen* A Generic Ranking Function Discovery Framework by Genetic Programming for Information Retrieval // Inf. Process. Manage. 40, 4 (May 2004), 587-602.

[15] *Варфаломеева А. А.* Методы структурного обучения для построения прогностических моделей // `http://www.machinelearning.ru/wiki/images/f/f2/Varfolomeeva2013Diploma.pdf`

[16] Trec conference // `https://trec.nist.gov/`

[17] K-mean algorithm. Sklearn implementation // `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`