

Прогнозирование ранжирующих моделей для систем информационного поиска.*

Поповкин А. А., Романенко И. И.

romanenko.ii@phystech.edu, popovkin.aa@phystech.edu

Московский физико-технический институт (государственный университет), Москва

Рассматриваются методы порождения нелинейных моделей для задач регрессии. Исследуется пространство моделей, представимых в виде суперпозиции математических примитивов, предлагаемых экспертами. Предлагается метод, прогнозирующий структуру оптимальной модели для задачи регрессии. Структура модели представляется как последовательности вершин при DFS обходе синтаксического дерева. Предложенный метод прогнозирует ранжирующую модель для систем информационного поиска на примере текстовой коллекции TREC. Приводится сравнительный анализ полученных результатов с уже известными моделями.

Ключевые слова: *информационный поиск, генетические алгоритмы, нейросети, структурное обучение.*

Введение

Решается задача ранжирования текстов по запросам пользователей. Решения данной задачи описаны в [1, 2, 3]. 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 [4] 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 [4], превосходят по качеству на коллекциях TREC модели из работ [1, 2, 3]. 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 [5, 6]. Производились поиски наилучших параметров генетического алгоритма. Как показано в [7, 8], оптимальный выбор операции кроссовера может существенно улучшить порождаемые модели.

Основным преимуществом генетического алгоритма является гибкость порождаемых им моделей, что позволило в [9, 10] перейти к представлению ранжирующей модели ее деревом синтаксического разбора. Однако генетический алгоритм, описанный в [9, 10], 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 [11], где благодаря использованию регуляризации функционала качества моделей, удается добиться улучшения разнообразия порождаемых функций, что ведет к повышению качества итоговой модели.

В работе [12] 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.

В данной работе предлагается альтернативная реализация генетического алгоритма со следующими изменениями: для увеличения разнообразия популяции предлагается каждые 5 эпох добавлять в популяцию случайную модель, в качестве мутаций изменять не функцию в узле, а целое поддерево, кроссовер проводится аналогичным образом, два объекта в популяции обмениваются случайными поддеревьями.

В качестве основной новации рассматривается развитие идеи предсказания промежуточной мета-модели. Предлагается кластеризовать документы по значениям tf-idf подсчитанным по корпусу текста, исходя из предположения, что текст разбивается на кластеры, причем внутри кластера документы в при ранжировании будут находится близко.

Работа построена следующим образом. Сначала ставится базовая постановка задачи прогнозирования модели, по исходной коллекции документов и запросов. Затем задача уточняется при кластеризации документов и создание мета-модели, являющейся линейной суммой моделей на кластерах. В конце проводится вычислительный эксперимент и сравниваются полученные результаты с уже известными моделями.

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 $r(d, q) \rightarrow \{0, 1\}$, определенная экспертами, и показывающая, является ли данный документ d релевантным для запроса q (1 — является).

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

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

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

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 \mathbf{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).

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

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

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

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

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

Постановка задачи на кластерах документов

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

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

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

Определим ранжирующую функцию на кластерах:

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

Оптимизацию весов W будем производить при помощи поиска перебора.

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 объектов популяции используется обмен случайных двух узлов в деревьях суперпозиций.

Алгоритм 1 Создание ранжирующей функции для коллекции документов

Вход: N_{epoch} , \mathbf{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}_i

▷ Мутировать случайную суперпозицию из \mathcal{T}_i

▷ Отранжировать популяцию \mathcal{T}_i согласно метрике MAP

▷ Учитывая регуляризацию по числу вершин

▷ Выбрать наилучшие суперпозиции, составить из них популяцию \mathcal{T}_{i+1}

▷ Увеличить число пройденных эпох

пока $epoch \neq N_{epoch}$

Используемые данные

Для обучения и тестирования модели используется коллекция текстовых документов TREC [13]. В частности коллекции Trec-5 – Trec-8. Для каждой коллекции представлены набор запросов, ранжирование документов, проведенное 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`'а. Для подсчета непосредственных значений `tf` использовался `CountVectorizer`, значение `idf` вычислялось непосредственно согласно формулам выше.

Алгоритм 2 Вычисление `tf – idf` по корпусу документов

Вход: `C`

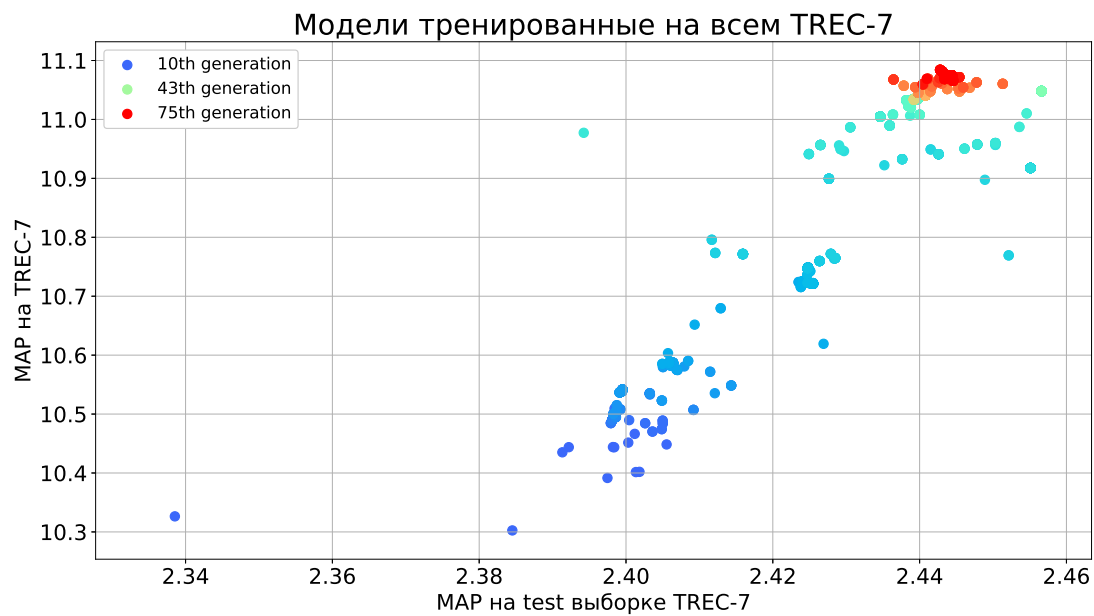
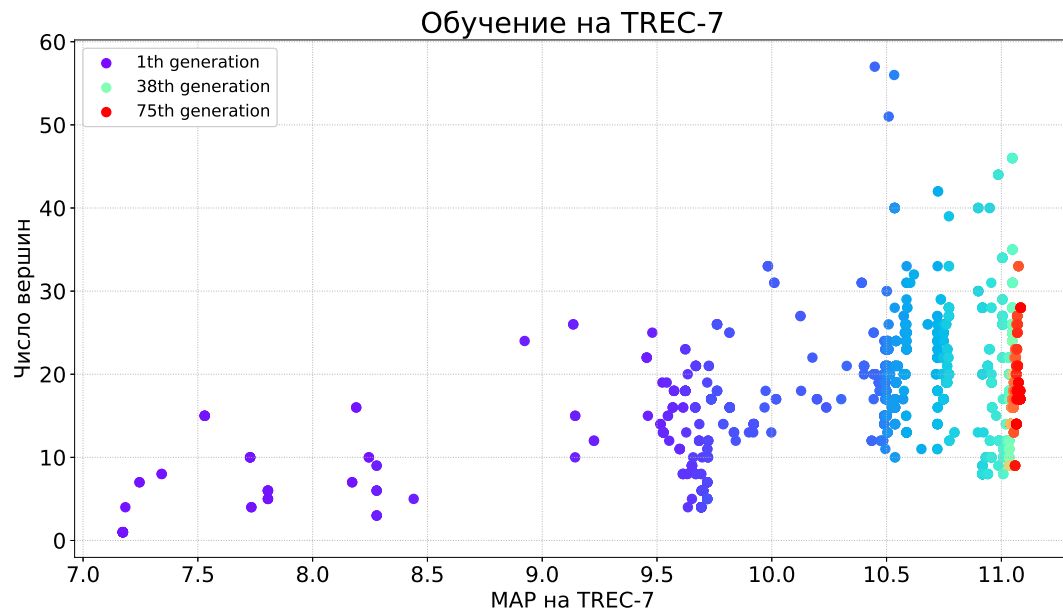
Выход: `tf – idf`

для каждого слова в коллекции документов `C`

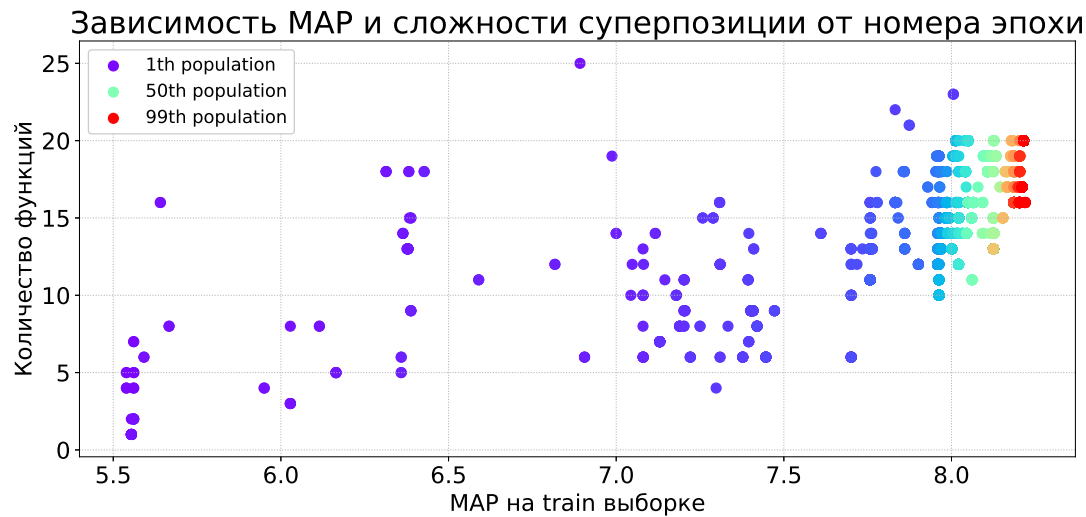
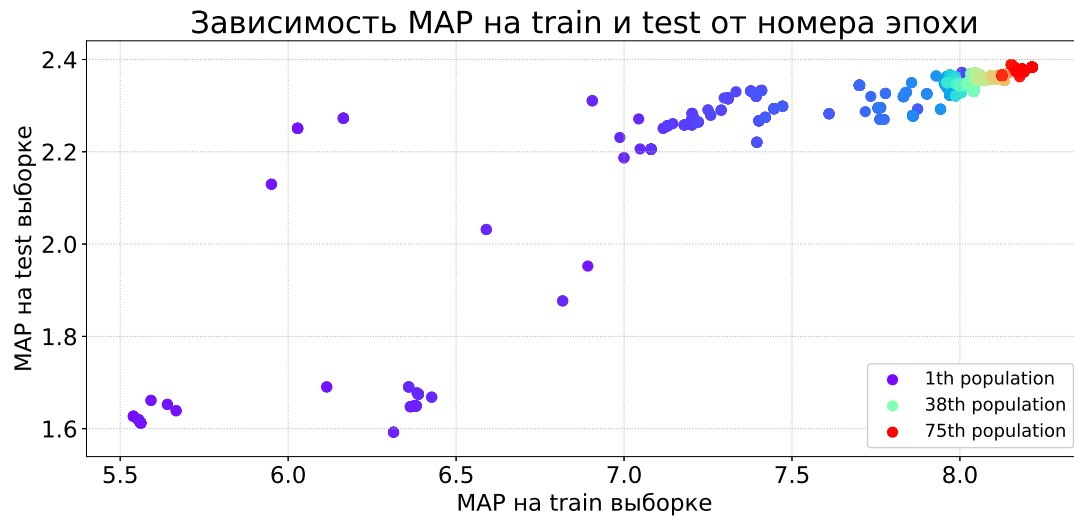
- ▷ Удалить знаки препинания и служебные символы
 - ▷ Разделить текст на слова
 - ▷ Удалить стоп-слова
 - ▷ Выполнить стемминг каждого слова
 - ▷ Вычислить значения `tf-idf` по матрице слов для документов
-

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.



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.

Экспертные функции:

$$f_1 = e^{\sqrt{\log(1 + \frac{tf}{idf})}}$$

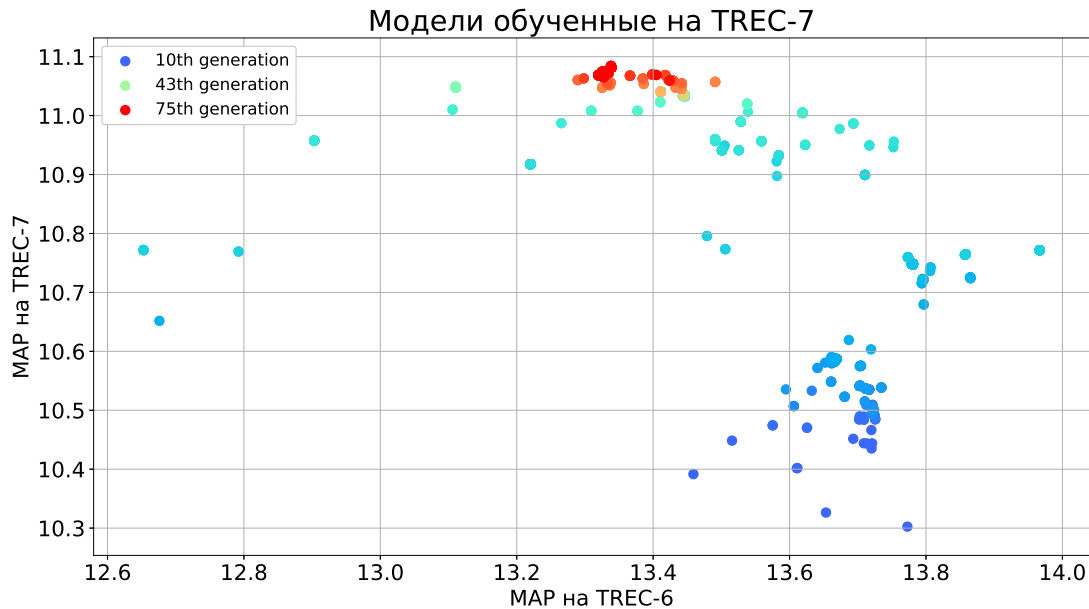
$$f_2 = \sqrt[4]{\frac{tf}{idf}}$$

$$f_3 = \sqrt{idf + \sqrt{\frac{tf}{idf}}}$$

Найденные наилучшие функции :

$$h_5^* = \log\left(1 + \frac{\log(1 + \log(1 + \log(1 + \sqrt{\log(1 + tf) - \sqrt{idf}})))}{2 \cdot idf}\right)$$

$$h_6^* = \sqrt{\frac{\sqrt[4]{tf}}{2 \cdot idf}}$$



$$h_7^* = \frac{\sqrt[8]{\log(1 + tf)}}{idf}$$

Результаты при сравнении на корпусах TREC-5, TREC-6, TREC-7.

Superposition	TREC-5	TREC-6	TREC-7
Функции сообщества			
f_1	8.785	13.715	10.038
f_2	8.908	13.615	9.905
f_3	8.908	13.615	9.905
Найденные наилучшие функции			
h_5^*	9.537	13.762	10.584
h_6^*	8.903	13.967	10.771
h_7^*	8.526	13.424	11.060

Вычислительный эксперимент при кластеризации данных

Так как кластеризация данного объема данных достаточно трудоемкая задача, был использован простой алгоритм K-means [14]. Количество кластеров, на которые разбивалась выборка было выбрано <TODO>.

Результаты при кластеризации корпуса <TODO>.

Заключение

<TODO>

Литература

- [1] *Salton, Gerard and McGill, Michael J.* Introduction to Modern Information Retrieval // McGraw-Hill, Inc., New York, NY, USA, 1986
- [2] *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.

- [3] *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.
- [4] *P. Goswami, S. Moura, E. Gaussier, M.-R. Amini, F. Maes* Exploring the space of ir functions // ECIR'14, 2014, pp. 372–384.
- [5] *Goldberg, David E.* Genetic Algorithms in Search, Optimization and Machine Learning // Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
- [6] *Koza, John R.* Genetic Programming: On the Programming of Computers by Means of Natural Selection // MIT Press, Cambridge, MA, USA, 1992.
- [7] *Vrajitoru, Dana* Crossover Improvement for the Genetic Algorithm in Information Retrieval // Inf. Process. Manage. 34, 4 (July 1998), 405–415.
- [8] *Gordon, M.* Probabilistic and Genetic Algorithms in Document Retrieval // Commun. ACM 31, 10 (October 1988), 1208–1218.
- [9] *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.
- [10] *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.
- [11] *Kulunchakov A. S., Strijov V. V.* Generation of simple structured IR functions by genetic algorithm without stagnation // <http://strijov.com/papers/Kulunchakov2014RankingBySimpleFun.pdf>
- [12] *Варфаломеева А. А.* Методы структурного обучения для построения прогностических моделей // <http://www.machinelearning.ru/wiki/images/f/f2/Varfolomeeva2013Diploma.pdf>
- [13] Trec conference // <https://trec.nist.gov/>
- [14] K-mean algorithm. Sklearn implementation // <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>