Actualization of Query Suggestions using Query Logs

Alisa Strizhevskaya, Alexey Baytin, Irina Galinskaya, Pavel Serdyukov
Yandex
16, Leo Tolstoy str.
Moscow, Russia
{amneziya, baytin, galinskaya, pavser}@yandex-team.ru

ABSTRACT

In this work we are studying actualization techniques for building an up-to-date query suggestions model using query logs. The performance of the proposed actualization algorithms was estimated by real query flow of the Yandex search engine.

Categories and Subject Descriptors:
H.3 [Information Storage and Retrieval]: H.3.3 Information Search and Retrieval

General Terms:
Algorithms, Measurement, Experimentation.

Keywords:
Query suggestions, Time series, Prediction algorithms.

1. INTRODUCTION.

Search suggestion is a tool that helps a search engines' user to type less while formulating a query. In response to the given user-input a search suggestions mechanism provides a list of possible prolongations. The aim is to suggest the correct prolongation by the shortest possible input prefix. On the other hand, each user query reflects some information intent, and intents are known to change over time. Therefore, search queries display some temporal dynamics, studied, for example, in [6]. By queries' actuality we mean the level of its interest for todays' user. For example, the actuality of the query "Halloween" grows all over October, drops in the beginning of November and stays static till next autumn.

Most of the today's query suggestion models operate with query logs [1] or static document collections [2]. That means that they, strictly speaking, build suggest models intended for todays' user, using past data, as aggregated query logs are available with a certain delay and the suggestion model is built off-line. In this paper a day is meant to be atomic time unit, but the model may be generalized to any other. Queries actuality is thought to be constant during the day, but the logs are available up to the day before. So we have to predict todays' queries' interest, knowing the dynamics of its interest during the past period. In order to smooth out that delay we propose to use linear time series prediction algorithms to actualize our suggestions.

To get a list of suggestions, a suggest server gets all queries from the dictionary, starting with the given prefix, sorts them by their frequencies and retrieves top results. In these terms, the goal of actualization boils down to the time series prediction: for each query we have a time series of its frequency during a certain period of time. Using this information we make a prediction of the query frequency to eliminate lag in data. However, there is a certain difference with the classical time series prediction problem. We are not interested in the precise query frequencies – our main goal is to obtain a realistic ranking by the predicted query frequencies. By ideal ranking we imply ranking made by real frequencies of queries on test day. Therefore, the performance of the algorithms will be evaluated with ranking metrics.

The contribution of this paper is the following. First, to the best of our knowledge, this is work is the first starting and addressing the problem of actualization of search suggestions. Second, we experimented with several time series prediction algorithms performing on the full production query stream of Yandex search engine and discovered that exponential smoothing algorithms are the best for this task.

2. ACTUALIZATION

The statement of the problem is as follows. For each query is given a time series of its daily frequencies during the certain period $\{f_1...f_N\}$. We must make a prediction for the next days' frequency – y_{N+1}. These predicted frequencies will be used for ranking of queries for each given prefix.

The web search engine's day stream (i.e. total amount of queries asked per day $stream_i = \sum_{query} freq_i(query)$) is known to change during a week: on weekends it decreases up to 20%. This effect adds an additional seasonal component to frequency time series of queries that is not explained by changes of actuality. This makes the problem more complicated as long as with changes in actuality we will have to predict changes in our stream. However, these particular stream changes are common for all queries and does not affect their relative relevances. Therefore we will eliminate stream component by normalizing day frequencies by day streams: $prob_i(query) = freq_i(query)/stream_i$. So, value $prob_i(query)$ is a probability of $query$ to be asked on $i-th$ day. We also smoothed up probabilities by convolving them with a gaussian (gauss filtering) to eliminate noise in data: $\overline{prob(query)} = prob(query) * \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$.

We would like to emphasize, that those transformations are made only for training data (i.e for the first N values). The evaluated prediction algorithms are:

Mean over the entire period is taken as the *baseline* prediction algorithm. We assume that it is well suited for

Algorithm	NDCG@3	AP@3	Prec@3
Brown	0.890(0.96%)	0.576(3.38%)	0.712(2.02%)
Holt	0.887(0.61%)	0.571(2.57%)	0.708(1.45%)
HoltWin	0.886(0.56%)	0.574(2.97%)	0.708(1.46%)
Autocorr	0.883(0.20%)	0.562(0.91%)	0.702(0.58%)
Mean	0.882	0.557	0.698
ARIMA	0.873(-0.93%)	0.548(-1.57%)	0.690(-1.14%)

Table 1: Results on top-3.

stable queries which are the most part of the dictionary, but fails to make good predictions for the queries that need to be actualized.

Autocorrelation model. Analysis of sample queries' behavior showed, that a considerable part of user queries is almost strictly periodical. For example, query "Raiffaisen bank" is stable during the week and has frequency dips on sundays. This model approximates each particular query time series as a periodical function. For every time series we calculate autocorrelation function and study its peaks, except the zero shift peak. We pick the shift of the peak with the largest magnitude and consider it to be queries' period. $prediction_{N+1} = freq_{N+1-period}$

Exponential smoothing (Brown's model). Exponential smoothing model without trend and seasonality [4].

Holt's model. Exponential smoothing considering linear trend [5].

Holt Winters multiplicative model. Exponential smoothing considering seasonality and linear trend. Seasonality period will be evaluated in the same way as in Autocorrelation model [7].

ARIMA(0,2,1). Auto regressive and integrated moving average model. This particular model configuration performs linear exponential smoothing [3].

Last four algorithms have free parameters that define models' sensitivities. We will adjust these parameters for every particular query' time series using the training data for that query. So, for $i = \overline{0,2}$ we use $freq_1...freq_{N-1-i}$ as an input and tune models' parameters by minimizing the least average error between the actual and predicted query frequency on the day_{N-i}. We take 3 days for training in order to avoid overlearning on outliers. Then we use these parameters for the test predictions on the day_{N+1}.

3. EXPERIMENTS

We will evaluate the algorithms by ranking metrics. We cut each query on $length(query) - 1$ prefixes and thereby obtain a list of all unique prefixes possible in the dictionary. For each of these prefixes we compare two posting tops of the same length: one – ranked by predicted probabilities, another – ranked by real frequencies. We will evaluate 3 metrics:

Precision and MAP. By relevant suggestions we imply queries that appear in the posting top, ranked by real frequencies.

NDCG Here the relevance gain of the $i - th$ position will be equal to its real frequency of the query on this position. In these terms, the ranking by real frequencies will be ideal and have maximal DCG.

The experiments will be held on 31 day logs. For the first 30 dates we calculate $stream_i$ and make prediction for every query appeared during the first 30 days more than 10 times. The last, $freq_{31}(query)$, is used to build the ground truth ranking of queries. After the prediction is performed, the set of queries is cut on all possible prefixes, and for every prefix we form a query posting list, ranked by predicted probabilities.

By default, Yandex search suggestions display at most 10 queries. On the other hand, an average user does not read all 10 suggestions and continues typing even if an intended query has already appeared in the tail of the posting. We performed an analysis of positions of clicked suggests during a week. It appeared that a suggest click was in 51.15% cases on the first position, in 20.8% on the second position, in 10.3% on the third. 17.8% are left for the rest seven positions. It means that the most important posting length is 3. Thereby, we will make evaluations on prefixes, whose full postings lengths are longer or equal to 3. And for each such prefix we will evaluate AP@3, Precision@3 and NDCG@3 by comparing top-3 of ideal and predicted postings on this prefix. In our case a set of 10,799,678 queries produced 83,795,381 unique prefixes, where 5,100,416 prefixes had postings longer than 3. The results are shown in table Table 1, differences significant to the baseline ($p < .05$) are shaded. Brown's model made better predictions than the baseline on 80% prefixes.

4. CONCLUSIONS

The results show that exponential smoothing algorithms lead on all considered metrics. The Brown's model appeared to be the best because it averages out data during the entire period and, therefore, performs good on stable queries that represent the majority among all unique queries. On the other hand, the exponential weight function makes it consider recent past in the first place, so it is sensible to the changes in actuality. The errors revealed that each of the algorithms has its own range of application. For example, ARIMA, performing fine on upper-medium frequency queries with a trend, fails to make good predictions on low frequency and periodical queries. Therefore, future work includes precise determination of these ranges of application of algorithms and using preliminary classification for queries. This classification will also be useful for detecting bursty queries, which appear to be the most problematic class for all considered algorithms. They may require some special prediction techniques and optimizations. This will also be a point of future study.

5. REFERENCES

[1] R. Baeza-yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *In International Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT), Creete*, pages 588–596. Springer, 2004.

[2] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. SIGIR '11, pages 795–804, New York, NY, USA, 2011. ACM.

[3] G. E. P. Box, G. C. Reinsel, and G. M. Jenkins. *Time series analysis : forecasting and control.* Prentice-Hall, Englewood Cliffs, NJ, 1994.

[4] R. Brown. *Smoothing, forecasting and prediction of discrete time series.* Prentice-Hall, Englewood Cliffs, NJ, 1963.

[5] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1), 2004.

[6] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. WSDM '11, pages 167–176, New York, NY, USA, 2011. ACM.

[7] P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 1960.