# Применение LSH для обнаружения похожих текстовых сообщений

С.Н. Середа, Р.В. Шарапов

*Муромский институт (филиал) ФГБОУВО «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»*
*602264, г. Муром, Владимирской обл., ул. Орловская, 23*
*E-mail: mivlgu@mail.ru*

*Работа посвящена вопросам применения LSH для обнаружения похожих текстовых сообщений. Суть LSH заключается в подборе хеш-функций для информационных объектов таким образом, чтобы похожие объекты с высокой степенью вероятности попадали в одну группу. В LSH похожие сообщения представляются похожими хешами небольшой размерности. Это достигается благодаря чувствительности LSH к местоположению информации. При этом соседние точки помещаются в один и тот-же хеш. Благодаря небольшим размерам, LSH хеш функции могут использоваться как сигнатуры принимаемых сообщений.*
*The work is devoted to the application of LSH to detect similar text messages. The essence of LSH is to select hash functions for information objects in such a way that similar objects with a high degree of probability fall into the same group. In LSH similar messages are represented by similar small hashes. This is achieved due to the sensitivity of the LSH to the location of the information. In this case, neighboring points are placed in the same hash. Due to their small size, LSH hash functions can be used as signatures of received messages.*

## Введение

LSH или Locality-sensitive hashing представляет собой вероятностный метод снижения размерности многомерных данных, получивший в последние годы большую популярность [1, 2].

Суть LSH заключается в подборе хеш-функций для информационных объектов (текстовых сообщений, изображений) таким образом, чтобы похожие объекты с высокой степенью вероятности попадали в одну группу.

LSH служит одним из способов борьбы с так называемым «проклятием размерности», когда при росте размерности исходных многомерных данных поиск по индексу ведет себя хуже чем последовательный просмотр. LSH позволяет создавать структуру для быстрого вероятностного поиска среди n-мерных векторов, похожих на искомый вектор. Она отображает множество точек в многоразмерном пространстве в хеш-таблицу.

LSH обладает следующими преимуществами:

1) простота использования;

2) строгая теория, подтверждающая хорошую производительность алгоритма;

3) LSH совместим с любой нормой $L_p$ при $0 < p \leq 20 < p \leq 2$. LSH используют с расстоянием Хэмминга, косинусным коэффициентов, манхэттенским расстоянием, евклидовой метрикой [3].

При традиционном хешировании два похожих сообщения, имеющих небольшие отличия, будет представлены значительно отличающимися хешами. В LSH, напротив, похожие сообщения представляются похожими хешами небольшой размерности. Это достигается благодаря чувствительности LSH к местоположению информации. При этом соседние точки помещаются в один и тот-же хеш.

**Обнаружение похожих текстовых сообщений с помощью LSH**

Простейшим способом построения LSH является бинарная выборка [1]. Такой подход использует расстояние Хэмминга для n-мерных бинарных векторов $\{0,1\}^n$. Тогда хэш-функциями $F$ будет являться проекции точек на каждую из $n$ координат:

$$F=\{h:\ \{0,1\}^n \rightarrow \{0,1\} \mid h(x)=x_i \text{ для некоторых } i\in\{1,\ldots,n\}\},$$

где $x_i$ – $i$-я координата $x$.

При этом случайная функция $h$ для $F$ просто выбирает случайный бит от входной точки [1].

Расстояние Хэмминга вычисляется по формуле [4]:

$$d_{ij} = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|.$$

Пусть словарь для коллекции документов содержит 100 000 слов. Тогда каждому документу можно поставить в соответствие 100 000 битный вектор, каждый элемент которого отражает присутствие или отсутствие i-го слова из словаря в документе (что соответствует классической булевой модели информационного поиска). Такой вектор, кстати, будет сильно разреженным.

Создадим для такого 100 000 битного вектора 256 битную LSH функцию. Для этих целей необходимо сгенерировать 256 случайных 100 000 битных векторов, которые будут являться основой для LSH-функции.

Для получения LSH-функции необходимо вычислить скалярное произведение вектора документа и каждого из 256 полученных ранее случайных векторов. Далее на основе скалярных произведений генерируется искомый 256-битный хеш по следующим правилам:

- если скалярное произведение вектора документа и случайного вектора больше нуля, то в соответствующий бит хеша записывается единица,
- если скалярное произведение вектора документа и случайного вектора меньше нуля, то в соответствующий бит хеша записывается ноль.

Таким образом происходит аппроксимация угла входного вектора в векторном пространстве и появляется возможность вычислить косинусный коэффициент двух хешей, сгенерированных данной LSH-функцией. Косинусный коэффициент двух текстовых документов (сообщений) будет пропорционален расстоянию Хэмминга между их LSH-хешами.

Существуют другие варианты вычисления хеш функций. Известность получили MinHash [5], Nilsimsa Hash, TLSH [6], SimHash [7] и т.д. Исследования показывают, что вектора миллионных размерностей с помощью LSH неплохо представляются 512 битными хеш функциями.

В настоящее время существуют сотни свободно-распространяемых реализаций LSH на различных языках программирования, что позволяет легко интегрировать его в проекты любого уровня сложности.

**Заключение**

LSH может с успехом использоваться при обнаружении похожих текстовых сообщений. Благодаря небольшим размерам, хеш функции могут использоваться как сигнатуры принимаемых сообщений. При этот, нет необходимости производить сравнение самих сообщений между собой – при поступлении нового сообщения высчитывается его сигнатура (хеш) и дальнейшее сравнение производится только с использованием таких сигнатур.

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00692.*

**Литература**

1. Indyk P., Motwani R. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. STOC98: The 30th Annual ACM Symposium on Theory of Computing. 1998. 604-613.
2. Rajaraman A., Ullman J. Mining of Massive Datasets. Cambridge University Press, 513 p.
3. Slaney M., Casey M. Locality-Sensitive Hashing for Finding Nearest Neighbors. IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 128-131, March 2008.
4. Hamming R.W. Error detecting and error correcting codes. The Bell System Technical Journal, vol. 29, no. 2, pp. 147-160, April 1950.
5. Broder A. On the resemblance and containment of documents. Compression and Complexity of Sequences: Proceedings, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997.
6. Oliver J., Cheng C., Chen Y. TLSH - A Locality Sensitive Hash. Proceedings. 4th Cybercrime and Trustworthy Computing Workshop, CTC 2013. P. 7-13.
7. Charikar M. Similarity estimation techniques from rounding algorithms. Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002. P. 380-388.