Seminar paper on Probabilistic Near-Duplicate Detection Using Simhash

Arne Beer, MN 6489196, University of Hamburg

08.07.2019

1 Introduction

In the age of the modern Internet, which depends in large parts on crawlers and proper document detection and duplication elimination, near duplicate document detection becomes a necessity. Real time detection of which website has already been visited and whether a website is new or just has been edited are important tasks during crawling [2].

Standard hashing functions are often inefficient and operate in $O(n^2)$ space requirements for RAM and computing time [1]. At the same time, the size of available documents grow steadily. Google's website index alone has multiple hundreds of billions of webpages and over 100 Petabyte in size, according to their information website [**info:google_stats**].

This paper attends to the paper *Seminar paper on Probabilistic Near-Duplicate Detection Using Simhash* [4]. The authors of [4] design a new algorithm that allows significantly faster online document comparison and reduced RAM requirements, for a small percentage of recall loss.

In the first chapters, the basics for understanding this topic will be explained. Afterwards the proposed algorithm will be looked at and the authors' findings will be discussed.

2 Conventional Hashing

Hashing is a technique, which is used to map data of an arbitrary size to a fingerprint with some fixed size. This procedure could be seen as a function $f(i) \rightarrow j$, which produces a value j from from any value i, where $j \in H$ and H is the set of values of the fixed length s with $s \in \mathbb{N}$. Well-known hash functions are, for instance, *md5* or *sha265*. These hashing functions are commonly used to check whether two files are absolutely identical or, for instance, to verify that a file has not been corrupted during transport. This is possible, since these hashing functions are designed to flip half of the output hash bits on average, if an input bit changes [1]. Without this property it would be easier to change the input without the hash signature being modified. This would allow malicious third parties to, for instance, change code in a binary, without users being able to detect the change with the help of this hash and would require a full byte level comparison between the original and the copied file to verify its integrity.

If, on the other hand, one's goal is to find near duplicates, which are identical for the most part, but sometimes only differ by a few bits or bytes, this hashing approach immediately becomes useless, due to this property. Due to the need for a hashing algorithm, that creates a fingerprint based on the features and structure of the input data, *simhash* has been created.

2.1 Simhash

[3]

2.2

2.3 Achievements of Session Juggler

3 Impact in the scientific community

4 Relevance as of 2018

References

[1] A.G. Konheim. *Hashing in Computer Science: Fifty Years of Slicing and Dicing*. July 2010. Chap. 8. DOI: 10.1002/9780470630617.

[2] Hsin-Tsang Lee et al. "IRLbot: Scaling to 6 Billion Pages and Beyond". In: *ACM Transactions on the Web (TWEB)* 3 (Jan. 2008), p. 8. DOI: 10.1145/1541822.1541823.

[3] Caitlin Sadowski and Greg Levin. "SimHash: Hash-based Similarity Detection". In: Dec. 2007. DOI: 10.1.1.473.7179.

[4] Sadhan Sood and Dmitri Loguinov. "Probabilistic Near-Duplicate Detection Using Simhash". In: Oct. 2011, pp. 1117–1126. DOI: 10.1145/2063576.2063737.