n-gram Cache Performance in Statistical Extraction of Relevant Terms in Large *Corpora*[⋆]

Carlos Goncalves[1,2][0000−0001−9113−6269], Joaquim F. Silva[2][0000−0002−5223−1180], and Jose C. Cunha[2][0000−0001−6729−8348]

[1] Instituto Superior de Engenharia de Lisboa cgoncalves@deetc.isel.pt
[2] NOVA Laboratory for Computer Science and Informatics {jfs,jcc}@fct.unl.pt

Abstract. Statistical extraction of relevant n-grams in natural language *corpora* is important for text indexing and classification since it can be language independent. We show how a theoretical model identifies the distribution properties of the distinct n-grams and singletons appearing in large *corpora* and how this knowledge contributes to understanding the performance of an n-gram cache system used for extraction of relevant terms. We show how this approach allowed us to evaluate the benefits from using Bloom filters for excluding singletons and from using static prefetching of nonsingletons in an n-gram cache. In the context of the distributed and parallel implementation of the LocalMaxs extraction method, we analyze the performance of the cache miss ratio and size, and the efficiency of n-gram cohesion calculation with LocalMaxs.

Keywords: Large *corpora*· Statistical extraction· Multiword terms· Parallel Processing· n-gram cache performance· Cloud computing

1 Introduction

Multiword expressions in natural language texts are n-grams (sequences of $n \geq 1$ consecutive words). Statistical extraction of relevant expressions, useful for text indexing and classification, can be language-independent. Thus it can be included in initial stages of extraction pipelines, followed by language-specific syntactic/semantic filtering. The increased availability of large *corpora* [1,2] due to the Web growth challenges statistical extraction methods. We focus on n-gram distribution models and parallel and distributed tools for extracting relevant expressions from large *corpora*. LocalMaxs [3,4], a multiphase statistical extraction method, has a 1^{st} phase for collecting n-gram frequency statistics, a 2^{nd} phase for calculating an n-gram cohesion metric, and a 3^{rd} phase for applying an n-gram relevance filtering criterion. The computational complexity of methods as LocalMaxs depends on n-gram distribution properties. Thus we proposed [5] a theoretical model predicting the n-gram distribution as a function of *corpus* size and n-gram size ($n \geq 1$), validated empirically for estimating the numbers of distinct n-grams, $1 \leq n \leq 6$, with English and French *corpora*

[⋆] Acknowledgements to FCT MCTES and NOVA LINCS UID/CEC/04516/2019

from 2 Mw (10^6 words) to 1 Gw (10^9 words) [6]. It allows to identify how the numbers of distinct n-grams tend asymptotically to *plateaux* as the *corpora* grow toward infinity. Due to the large numbers of distinct n-grams in large *corpora*, the memory limitations become critical, motivating optimizations for space efficient n-gram data structures [7]. We pursue an orthogonal approach using parallel computing for acceptable execution times, overcoming the memory limitations by data partitioning with more machines, and using data distribution for scalable storage of the n-grams statistical data [8,9]. Guided by the theoretical model estimates, we developed a parallel architecture for LocalMaxs: with an on-demand dynamic n-gram cache to keep the n-gram frequency data, used for supporting the cohesion calculations in the 2^{nd} phase; a distributed in-memory store as a repository of the n-gram global frequency values in the *corpus* and the cohesion and relevance values; and a workflow tool for specifying multiphase methods; supported by a distributed implementation with a configurable number of virtual machines. LocalMaxs execution performance for extracting relevant 2-grams and 3-grams from English *corpora* up to 1 Gw was shown scalable, with almost linear relative speed-up and size-up, with up to 48 virtual machines on a public cloud [8,9]. However, that implementation achieves low efficiency relative to a single ideal sequential machine because the on-demand dynamic n-gram cache is unable to overcome the communication overheads due to the n-gram references missing in the cache, requiring the remote fetching of the n-gram global frequency counts. To improve the n-gram cache efficiency, we discuss two new aspects, as extensions to the LocalMaxs parallel architecture. The first one consists in filtering the singleton n-grams. To evaluate this, we extend the theoretical model to predict the distribution of singleton n-grams, $1 \leq n \leq 6$, applying this to English *corpora* from a few Mw to infinity. Then we show that this singletons filtering with Bloom filters [10] leads to a reduction of the n-gram cache miss ratio, but it depends on the evolution of the numbers of singletons as the *corpus* size grows. The second improvement relies on the static prefetching of the n-grams statistical data into the cache. This, for a multiphase method, can be performed completely in the 1^{st} phase (collecting n-gram statistics), so that during a subsequent phase where the n-gram cache is used for cohesion metric and relevance calculation, there is no cache miss overhead. For LocalMaxs, this leads to a virtually 0% cache miss ratio for any *corpus* sizes. In the paper we discuss background (sec. 2), the theoretical model and the distribution of singleton n-grams (sec. 3), the two n-gram cache improvements (sec. 4) and the obtained results (sec. 5).

2 Background

Relevant expressions, e.g. “United Nations”, can be used to summarize, index or cluster documents. Due to their semantic richness, their automatic extraction from raw text is of great interest. Extraction approaches can be linguistic, statistical or hybrid [11,12]. Most of the statistical ones are language-neutral [13], using metrics as Mutual Information [14], Likelihood Ratio [15], Φ^2 [16]. Among the latter, LocalMaxs [3,4] extracts multiword relevant expressions [17].

LocalMaxs It relies on a generic cohesion metric, called "glue", (as SCP_f eq. (1) below; Dice [4]; or Mutual Information), and on a generic relevance criterion (as eq. (2) below), that, for a given input set of n-grams ($n \geq 2$), identifies the ones considered relevant, according to the strength of their internal co-occurrence:

$$SCP_f(w_1\cdots w_n)=\frac{f\left(w_1\cdots w_n\right)^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} f\left(w_1\cdots w_i\right)\times f\left(w_{i+1}\cdots w_n\right)} \tag{1}$$

where $f\left(w_1\cdots w_i\right)$ is the frequency of the n-gram $\left(w_1\cdots w_i\right)$, $i \geq 1$, in the *corpus*. The denominator has the frequencies of all "leftmost and rightmost sub n-grams" (that, for simplicity, are abbreviated as "sub n-grams") of sizes from 1 to $n-1$ contained in $\left(w_1\cdots w_n\right)$. E.g., considering the 5-gram "European Court of Human Rights", the sub n-grams whose frequencies are needed for the glue calculation are: the 1-grams, "European" and "Rights"; the 2-grams, "European Court" and "Human Rights"; the 3-grams, "European Court of" and "of Human Rights"; and the 4-grams, "European Court of Human" and "Court of Human Rights".

LocalMaxs Relevance Criterion Let $W=(w_1 \ldots w_n)$ be an n-gram and $g\left(.\right)$ a generic cohesion metric. Let $\Omega_{n-1}\left(W\right)$ be the set of $g\left(.\right)$ values for all contiguous $(n-1)$-grams within the n-gram W; Let $\Omega_{n+1}\left(W\right)$ be the set of $g\left(.\right)$ values for all contiguous $(n+1)$-grams containing n-gram W. W is relevant expression iff:

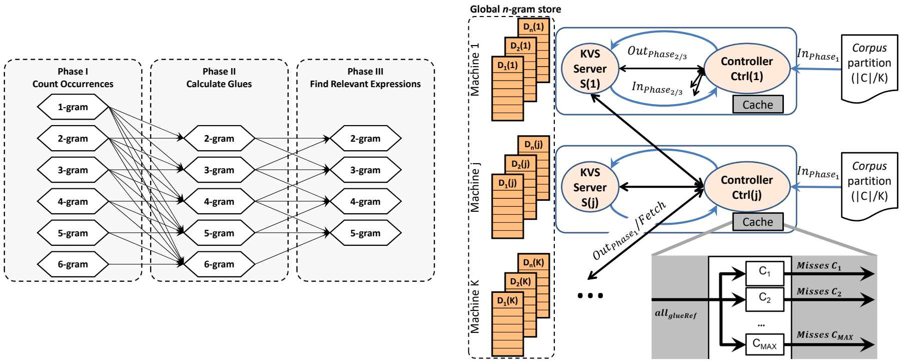
$$\begin{gathered}\forall_x \in \Omega_{n-1}\left(W\right), \forall_y \in \Omega_{n+1}\left(W\right) \\ \mathit{length}\left(W\right)=2 \wedge g\left(W\right)>y \quad \vee \quad \mathit{length}\left(W\right)>2 \wedge g\left(W\right)>\tfrac{x+y}{2}\end{gathered} \tag{2}$$

For the example $W=(\mathit{European\,Court\,of\,Human\,Rights})$, the sets are: $\Omega_{n-1}(W)=\{g\left(\mathit{European\,Court\,of\,Human}\right), g\left(\mathit{Court\,of\,Human\,Rights}\right)\}$; and $\Omega_{n+1}(W)=\{g\left(Y\right)\}$, such that $Y=(w_L\ W)$ or $Y=(W\ w_R)$ where symbols w_L and w_R stand for unigrams appearing in the *corpus*, and Y is the $(n+1)$-gram obtained from the concatenation of w_L or w_R with W.

Parallel LocalMaxs Architecture Figure 1a shows the logical dependencies of LocalMaxs for extracting relevant n-grams, $2 \leq n \leq 5$. For a given maximum n-gram size n_{MAX} the relevant n-grams, $2 \leq n \leq n_{MAX}$, are identified in the *corpus* in three phases: (1) counting all n-gram occurrences, $1 \leq n \leq (n_{MAX}+1)$; (2) calculating the glue ($g_{2\cdots(n_{MAX}+1)}$) for all distinct n-grams, $2 \leq n \leq (n_{MAX}+1)$; and (3) applying a relevance criterion to all distinct nonsingleton n-grams, $2 \leq n \leq n_{MAX}$. The workflow is executed [8,9] by a collection of virtual machines, each with one controller (for LocalMaxs functions: count, glue, relevance), one server (for storing the n-gram data), and local n-gram caches (Figure 1b).

In phase one, the n-gram counting is performed in parallel by different controllers acting on equal-size input *corpus* partitions. It generates the distinct n-gram tables, one for each n-gram size, containing the total counts of all the n-gram occurrences in the *corpus*. These tables, partitioned by n-gram hashing, are stored in a distributed collection of servers, thus supporting a repository of the global n-gram frequency counts in the *corpus* (in the end of phase one).

For K machines, each server $S(j)$ in each machine j: $1 \leq j \leq K$, keeps a local n-gram table $(D_i(j))$, for n-grams of size i: $1 \leq i \leq (n_{MAX}+1)$. The set of local n-gram tables $(1 \leq i \leq (n_{MAX}+1))$ within each server $S(j)$ is: $\{D_1(j), D_2(j), \cdots, D_i(j), \cdots, D_{n_{MAX}+1}(j)\}$. The set of distinct n-grams of size i in the *corpus* (D_i) is the union of the disjoint local n-gram tables $D_i(j)$ in all servers $1 \leq j \leq K$. In each machine (j), there is one controller $(Ctrl(j))$ co-located with one local server $S(j)$. Phase two input consists of a set of distinct n-grams whose glues must be calculated. These n-grams and their frequency counts are found, by each machine controller, in the local server n-gram tables. However, the frequencies of the sub n-grams required for glue calculation of each distinct n-gram must be fetched from the global distributed repository. So, in this phase the repeated sub n-gram references used by the glue calculations justify a per machine n-gram cache for each n-gram size (C_1, ..., $C_{n_{MAX}}$) (Figure 1b). Each local cache entry has the frequency of a distinct n-gram. In the end of phase two all the distinct n-gram entries in the global repository become updated with their glue values. The input to phase three, for each machine controller, consists of the local n-gram tables updated by phase two, used to evaluate the n-gram relevance, finally stored in the local n-gram table. At the end, for all tables $(D_i(j))$ of the global repository, each entry has: an unique n-gram identification, its global frequency, its glue value, and its relevance flag (yes/no). As the *corpus* data is unchanged during LocalMaxs execution, a static work distribution leads to a balanced load in all phases since the local table sizes are approximately equal, $|D_i(j)| \approx (|D_i|/K)$ (for each n-gram size i, $1 \leq i \leq n$; machine j, $1 \leq j \leq K$), and the controller input partitions are of equal sizes.



(a) Logical workflow (b) Work table and data partitioning

Fig. 1: Parallel LocalMaxs architecture

3 A Theoretical Model for n-gram Distribution

We review a theoretical model [5] for the efficient estimation of the number of distinct n-grams ($n \geq 1$), for any *corpus* size, for each given language. Here the model is extended for predicting the number of singleton n-grams.

Distinct n-grams By Zipf-Mandelbrot Law [18,19] and Poisson distribution:

$$f\left(r,c,n\right)=\left(\left(1+\beta\left(n\right)\right)^{\alpha(n)}\times f\left(1,c,n\right)\right)\times\frac{1}{\left(r+\beta\left(n\right)\right)^{\alpha(n)}}\tag{3}$$

where $f\left(r,c,n\right)$ is the absolute frequency of the r^{th} most frequent n-gram of size n in a *corpus* C of $c=|C|$ words. The most frequent n-gram of size n is ranked $r=1$ with frequency $f\left(1,c,n\right)$, and the least frequent n-gram of size n has rank $r=D\left(c,n\right)$, i.e., the number of distinct n-grams of size n for a *corpus* C. For each language, $\alpha\left(n\right)$ and $\beta\left(n\right)$ are approximately constant (in eq. (3)). As confirmed empirically, the relative frequency, $p_1\left(n\right)$, of the first ranked n-gram tends to be constant wrt the *corpus* size: $f\left(1,c,n\right)=p_1\left(n\right)\times c$. Thus, $\left(\left(1+\beta\left(n\right)\right)^{\alpha(n)}\times f\left(1,c,n\right)\right)$ in eq. (3) is constant for each *corpus* size, hence the frequency of each rank follows a power law with $\alpha\left(n\right)>0$. Let random variable X be the number of occurrences of n-gram w in rank r in a *corpus*, in language l, by Poisson distribution, the probability of w occurring at least once is:

$$Pr\left(X\geq 1\right)=1-e^{-\lambda}\tag{4}$$

where λ is the Poisson parameter, the expected frequency of n-gram w in that *corpus*. For each rank r, we have $\lambda=f(r,c,n)$. Thus, $Dist\left(l,c,n\right)$, the expected number of distinct n-grams of size n in a *corpus* of size c in language l, is:

$$Dist\left(l,c,n\right)=\sum_{r=1}^{v(n)}\left(1-e^{-f(r,c,n)}\right)=v\left(n\right)-\sum_{r=1}^{v(n)}e^{-\left(\left(\frac{1+\beta(n)}{r+\beta(n)}\right)^{\alpha(n)}\times\left(p_1(n)\times c\right)\right)}\tag{5}$$

For each n-gram size n there is a corresponding language n-gram vocabulary of specific size $v\left(n\right)$, which in our interpretation includes all different word flexions as distinct. The parameters α, β, p_1, v were estimated empirically for the English language, for 1-grams to 6-grams [5], using a set of Wikipedia *corpora* from 2 Mw to 982 Mw (Table 1). In Figure 2a, the curves for the estimates (Es) of the numbers of distinct n-grams ($1\leq n\leq 6$) are shown dotted and for the observed data (Obs) are filled, corresponding to a relative error ($Es/Obs-1$) generally below 1% [5]. Above well identified *corpus* size thresholds, for each n-gram size, the number of distinct n-grams reaches an asymptotic *plateau* determined by the finite vocabulary size, at a given time epoch. Any further *corpus* increase just increases the existing n-grams frequencies.

Number of Singleton n-grams From the Poisson distribution, the number of distinct n-grams with frequency $k\geq 0$ is estimated as:

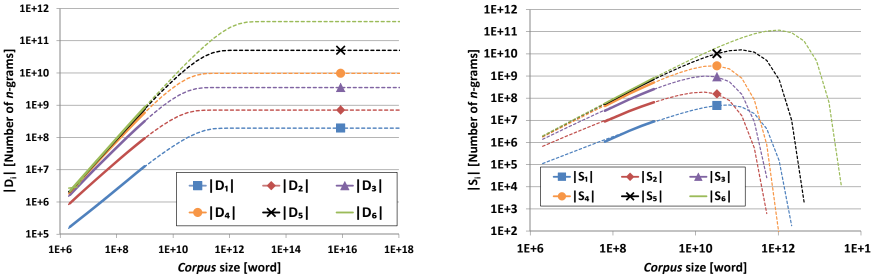
$$W(k,c,n)=\sum_{r=1}^{r=v(n)}\frac{\lambda_r^k\times e^{-\lambda_r}}{k!}=\sum_{r=1}^{r=v(n)}\frac{f(r,c,n)^k\times e^{-f(r,c,n)}}{k!}\tag{6}$$

where $\lambda_r=f\left(r,c,n\right)$. For k=1 it estimates the number of singletons (Figure 2b), $1\leq n\leq 6$. The number of singletons increases with the *corpus* size, as new ones

Table 1: Best α, β, v (number of n-grams) and p_1 for the English *corpora*

	unigrams	**bigrams**	**trigrams**	**trigrams**	**pentagrams**	**hexagrams**
$\boldsymbol{\alpha}$	1.3466	1.1873	0.9800	0.8252	0.8000	0.8000
$\boldsymbol{\beta}$	7.7950	48.1500	21.8550	0.4200	-0.4400	0.6150
$\boldsymbol{v}$	1.95×10^8	7.08×10^8	3.54×10^9	9.80×10^9	5.06×10^{10}	3.92×10^{11}
$\boldsymbol{p_1}$	0.05037	0.00827	0.00239	0.00238	0.00238	0.00067

keep appearing until a maximum, and vanishes gradually due to the vocabulary finiteness. Singletons keep a significant proportion of the distinct n-grams for a wide range: e.g., proportions fall below 80% only for *corpora* around 8 Mw, 1 Gw, 4 Gw, 16 Gw, 131 Gw, respectively, for 2-grams, 3-grams, 4-grams, 5-grams, 6-grams. Singleton 1-gram proportion is above 55% for *corpora* up to 16 Gw.



(a) Distinct n-grams (D_i), $1 \leq i \leq 6$

(b) Singletons n-grams (S_i), $1 \leq i \leq 6$

Fig. 2: Estimates (dotted) and empirical data (filled)

4 *n*-gram Cache System

Caching has been widely studied: in linguistics [20], Zipf distributions [21], Web search [22], or mining [23]. An n-gram cache is useful to statistical methods. In [9] a dynamic on-demand n-gram cache exploits repetitions in texts, reducing access overheads to a remote store in LocalMaxs 2^{nd} phase. Overheads were further reduced [9]: by an n-gram cache warm-up using combined metric calculations; and by using more machines, thus reducing the per machine number of n-gram misses (albeit non linearly) and the miss time penalty. That reduction is still not enough. Thus, we discuss two new improvements, validated experimentally for English *corpora* up to 1 Gw. Firstly (sec. 4.2), we filter the large proportions of singletons in the *corpus*, out of the n-gram cache, using Bloom filters [10]. In [24], alternatives for Bloom filters, caching and disk/in-memory storage were evaluated but focused on performance and scalability in text mining. Distinctively we developed an n-gram cache for $n \geq 1$ and analyzed Bloom filters efficiency depending on the numbers of singletons, from small *corpus* sizes up to infinity.

DOI: 10.1007/978-3-030-22741-8_6

Secondly (sec. 4.3), using static prefetching we achieved a 0% n-gram cache miss ratio.

An n-gram cache in LocalMaxs 2^{nd} ***phase*** For each glue calculation, references to sub n-grams are generated, which are submitted as cache input references to check if they are already in the local n-gram cache, otherwise they must first be fetched from the global n-gram repository. The set of references, $all_{glue_{g_n}Ref}\,(j)$, contains all sub n-gram occurrences for glue calculation (g_n) of the distinct n-grams of size n in table $D_n(j)$ in machine j. The set of distinct sub n-grams (sizes 1 to $(n-1)$), found within $all_{glue_{g_n}Ref}\,(j)$, is $D_{all_{1\cdots(n-1)}}(j) = D_{1_{in}D_2\cdots D_n}(j) \cup D_{2_{in}D_3\cdots D_n}(j) \cup \cdots \cup D_{(n-1)_{in}D_n}(j)$. Each set $D_{i_{in}D_n}$, $1 \leq i \leq (n-1)$, contains the distinct sub n-grams of size i, occurring within the n-grams in D_n table (eq. (1)). For a single machine, $D_{i_{in}D_n}$ is D_i, $1 \leq i \leq (n-1)$, the set of distinct n-grams of size i in the *corpus*. For multiple machines, each one handles the distinct sub n-gram references in its local tables ($D_2(j)$, $D_3(j)$, etc.).

4.1 Dynamic on-demand n-gram Cache

A dynamic on-demand n-gram cache, assumed unbound, is able to contain all the distinct sub n-grams of each local n-gram table. We analyzed the cold-start (first occurrence) misses behavior, for an initially empty cache. If there is not enough memory, the cache capacity misses are also handled. To reduce the cold misses overhead we built an n-gram cache warm-up [9] using combined glues: whereas the single glue calculation for 2-grams (g_2) only requires access to the 1-gram cache, for the combined glues of 2-grams up to 6-grams ($g_{2\cdots6}$) the 1-gram cache (C_1) is reused five times, the 2-gram cache (C_2) is reused four times, and so on for caches C_3, C_4, C_5. In a single machine, the global miss ratio (mr) of an unbound cache system with subcaches C_1 to C_5 used for glue $g_{2\cdots6}$, is:

$$mr = \frac{D_{all_{1\cdots5}}}{all_{glue_{g_2\cdots g_6}Ref}} = \frac{\sum_{i=1}^{5}|D_i|}{\sum_{i=2}^{6} 2 \times (i-1) \times |D_i|} \tag{7}$$

The miss ratio decreases with the *corpus* size and increases with the glue calculation complexity (n-gram size). Using the theoretical model for a single machine, we predicted the evolution of the miss ratio of the dynamic on-demand n-gram cache (Figure 3a) for glue $g_{2\cdots6}$: it varies from around 11%, for *corpus* size close to 10 Mw, to an asymptotic limit of around 1.5% in the *plateaux* (beyond 1 Tw). Results were experimentally validated for English *corpora* up to 1 Gw.

Effect of Multiple Machines $(K>1)$ Due to a multiplication effect of the nonsingletons (mostly the common ones e.g. "the", "and") cited by multiple distinct n-grams spread across multiple machines [9], the number of distinct sub n-grams for glue calculation in each machine is not reduced by a factor of K wrt the number of distinct n-grams in the *corpus*, unlike the number of cache references per machine that is reduced as $1/K$ compared to the case of a single machine. Thus, the per machine miss ratio of a dynamic on-demand n-gram

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_6

cache increases with K for each *corpus* size. Indeed we have shown [9] that, for each n-gram size n, the miss ratio follows a power trend: $mr(K) \propto K^{b(n)}$, $0<b(n)<1$.

4.2 Bloom Filters for Singletons in an On-demand n-gram Cache

Singletons can be filtered by Bloom filters [10], trained with the nonsingletons occurring in the *corpus*. For the majority of singletons the Bloom filter says: “definitely not in set”. For all the nonsingletons and a minority of singletons it says: “possibly in set”. The percentage of false positives is kept low enough by adequate Bloom filter implementation. During phase one of LocalMaxs each server (sec. 2) generates a Bloom filter for each local n-gram table. At the end of phase one, after the servers have updated all the n-gram frequencies, the filters were trained with all the nonsingletons. In the beginning of phase two, each machine controller gets a copy of the trained Bloom filters.

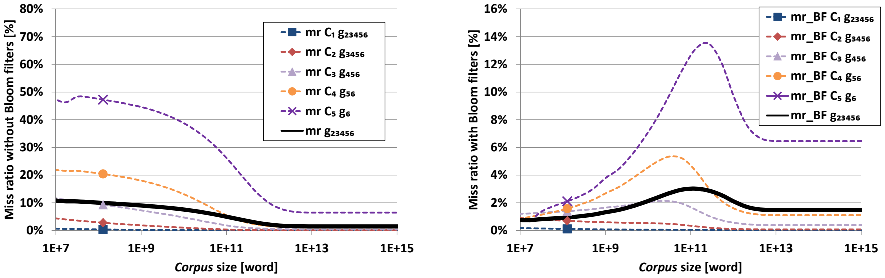
Single Machine Case (**K**=1) The proportion of the total number of singletons (S_{all}) wrt the total number of distinct n-grams in the *corpus* (D_{all}) is:

$$SF_{all} = \frac{S_{all}}{D_{all}} = \frac{\sum_{i=1}^{5} S_i}{\sum_{i=1}^{5} D_i} \tag{8}$$

also illustrating the SF_{all} ratio for the case of glue $g_{2\cdots6}$. Thus:

$$\frac{mr}{mr_{BF}} = \frac{\frac{D_{all}}{all_{glueRef}}}{\frac{(D_{all}-S_{all})}{all_{glueRef}}} = \frac{1}{1-SF_{all}} \tag{9}$$

where mr and mr_{BF} are, respectively, the miss ratio without and with Bloom filters. The case of glue $g_{2\cdots6}$ is illustrated in Figure 3 where the miss ratios of the individual caches C_1 to C_5 are shown (dotted), as well as the global miss ratio of the cache system $C_{1+\cdots+5}$ (filled). The curves result from the model predictions, and were experimentally validated for *corpus* sizes up to 1 Gw.



(a) Without Bloom filters (b) With Bloom filters

Fig. 3: Dynamic on-demand cache mr for K=1, glue $g_{2\cdots6}$. Different Y−scales

Due to the composition of the individual miss ratios of the caches C_1 to C_5, the miss ratio with Bloom filters (Figure 3b) for glue $g_{2\cdots6}$ is mostly dominated by the larger populations of the larger n-gram sizes. It varies from about 1% for the smallest *corpus* (about 10 Mw) to 1.5% in the *plateau.* The reduction, wrt not using Bloom filters, is due to the increased filtering effectiveness in handling the large global proportion of singleton n-grams, $1 \leq n \leq 5$. The miss ratio has a non monotonic behavior, with a single peak of about 3% at around 100 Gw, however for *corpora* until 1 Gw it remains always below about 1.3%.

Effect of Multiple Machines (K>1) The per machine miss ratio is:

$$mr_{BF}{=}\frac{\hat{D}_{all}-\hat{S}_{all}}{\hat{all}_{glueRef}}{=}\frac{\hat{NS}_{all}}{\hat{all}_{glueRef}}{=}\frac{NS_{K=1}\times K^{-b_{NS}}}{all_{glueRef}/K}{=}\frac{NS_{K=1}\times K^{1-b_{NS}}}{all_{glueRef}} \tag{10}$$

where $\hat{D}_{all}{=}\left(\sum_{j=1}^{K} D_{all_{1\cdots(n-1)}}(j)\right)/K$ is the per machine number of distinct sub n-grams of sizes 1 to n–1; $\hat{S}_{all}$ is the per machine number of singleton sub n-grams of sizes 1 to n–1. Due to their multiplicative effect with K (sec. 4.1), the number of per machine nonsingletons follows $\hat{NS}_{all} \propto K^{-b_{NS}}$, $0{<}b_{NS}{<}1$ (b_{NS} empirically determined). Thus mr_{BF} increases with K. Table 2 shows experimental values of the miss ratios without and with Bloom filters, for a LocalMaxs implementation using $K = 16$ machines in a public cloud [25], compared to a single machine case, for *corpora* of sizes 205 Mw and 409 Mw, when calculating glue g_{234}.

Table 2: Distinct n-grams, singletons, cache references (numbers of n-grams); Cache miss ratio (%) without/with Bloom filters

	$K = 1$		$K = 16$	
	$\lvert C\rvert = 205\ Mw$	$\lvert C\rvert = 409\ Mw$	$\lvert C\rvert = 205\ Mw$	$\lvert C\rvert = 409\ Mw$
$\hat{D}_{all_{1\cdots3}}$	$115,803,664$	$201,335,533$	$22,075,227$	$38,276,819$
$\hat{S}_{all_{1\cdots3}}$	$92,600,190$	$158,713,260$	$13,874,669$	$23,569,998$
$\hat{all}_{glueRef_{g_2\cdots g_4}}$	$1,222,207,756$	$2,236,879,374$	$76,386,941$	$139,802,828$
mr	9.47 %	9.00 %	28.90 %	27.38 %
mr_{BF}	1.90 %	1.91 %	10.74 %	10.52 %

Overall, Bloom filters lead to a reduction in the cache size and in the miss ratio, both determined by the singleton ratio (SF_{All}). As this ratio tends to zero the Bloom filter effect diminishes progressively.

4.3 n-gram Cache with Static Prefetching

Whenever one can identify the set of distinct n-grams in a *corpus* and their frequencies in a 1^{st} phase, one can anticipate their fetching into the n-gram cache before the execution starts in a 2^{nd} phase. The Fixed Frequency Accumulation

Set (`FAset`) FA for ensuring a static hit ratio $h_S\,(n, FA)$ is the minimal subset of distinct n-grams of a given size n, whose cumulative sum of frequencies is a percentage of the number of occurrences of the n-grams of size n:

$$h_S\,(n, FA) = \frac{\sum\limits_{ng \in FA} freq_{inCorpus}\,(ng)}{|Set_{All_n}|} \tag{11}$$

where ng is a distinct n-gram within the `FAset` and $freq_{inCorpus}\,(ng)$ is its frequency in the *corpus* C; and $|Set_{All_n}| = |C| - (n-1)$ for $n \geq 1$. When applying this concept to an n-gram cache one must consider, as the denominator of eq. (11), the set of cache input references $all_{glue}Ref_{n-gram}$ instead of the set Set_{All_n}. For glue g_2 of the 2-grams in the D_2 table, LocalMaxs requires access to all the subunigram occurrences (in a total of $all_{g_2}Ref_{1-gram} = 2 \times |D_2|$). The `FAset` to be loaded in cache C_1 is the subset of the elements in the set $D_{1_{in\,D_2}}$ (sec. 2) whose accumulated sum of frequencies of occurrences within the 2-grams of the D_2 table ensures a desired static hit ratio ($h_{S_{C_1}}$) for the 1-grams cache. For a combined glue, e.g. g_{234}, using caches C_1, C_2 and C_3 (1-grams, ..., 3-grams), let $freq_{in\ all_{g_{234}}Ref_{i-gram}}\,(ng)$ $(1 \leq i \leq 3)$ be the frequency of a distinct n-gram ng occurring in the set of cache input references $all_{g_{234}Ref_{i-gram}}$. To ensure a target static hit ratio h_S (or miss ratio mr_S) the `FAset` must enforce the following proportion of hits ($nbrHits$) wrt the total number of cache references ($|all_{g_{234}Ref}| = \sum_{i=1}^{3} all_{g_{234}Ref_{i-gram}}$, for glue g_{234}):

$$h_S = \frac{nbrHits}{|all_{glueRef}|} = \frac{\sum_{i=1}^{3} \sum\limits_{ng \in FAset} freq_{in\ all_{g_{234}}Ref_{i-gram}}(ng)}{|all_{g_{234}Ref}|} = 1 - mr_S \tag{12}$$

Options for selecting the distinct n-grams for the `FAset` are: i) All the distinct n-grams; ii) Only the nonsingletons; iii) A subset of the distinct n-grams. Option (i) seems the best but there is no need to include the singletons, which suggests option (ii). If there is not enough memory for all the nonsingletons in the cache, option (iii) must be taken ensuring the maximum number of hits per n-gram, under the existing memory constraints [17]. The LocalMaxs workflow (Figure 1a) allows to completely calculate the `FAsets` for each n-gram size in phase one, overlapped with the n-gram counting, using dedicated threads. As the machine allocation to LocalMaxs tasks in all phases is made before execution starts, one can also prefetch the `FAsets` into the corresponding machines in phase one. Thus the `FAset` calculation and prefetching times are hidden from the total execution time, as far as the additional thread overheads are kept small. In option (ii), by prefetching all distinct nonsingletons completely in phase one, the nonsingleton miss overheads in phase two are eliminated, leading to a 0% overall miss ratio.

Multiple Machines Case (K>1) The `FAset` size per machine decreases with K as the number of distinct sub n-grams per machine [17]. But, unlike the dynamic on-demand cache, the miss ratio with static prefetching can be kept constant wrt K by adjusting the per machine `FAset` according to the number of machines, e.g., for a 0% miss ratio, all the nonsingletons in the per machine distinct n-gram tables must be always included in the local `FAset`.

Experimental Results We compared the communication and glue calculation times of static prefetching of all nonsingletons *versus* on-demand caching. In each machine (j), phase two takes a total time $T_2\,(j)$ consisting of time components for: input $T_{input}\,(j)$; local glue calculation $T_{Glue}\,(j)$; sub n-gram fetch $T_{comm}\,(j)$; glue output $T_{output}\,(j)$. The input/output consists of local machine interactions between the co-located server and controller (Figure 1b), being the same in both cache cases. Table 3 shows the communication and glue times of g_{234} (machine average), for two *corpus* sizes, in LocalMaxs phase two [9,17] in a public cloud [25] with 16 machines (each 64 GB `RAM, 4 vCPU@1.5 GHz`).

Table 3: Glue and communication times $(min{:}sec) K{=}16$: Dynamic *vs.* Static cache

	$\hat{T}_{comm} + \hat{T}_{Glue}$	$\hat{T}_{RemoteFetch}$
	(Dynamic | Static)	(Dynamic | Static)
$\|C\| = 205\, Mw$	07:20 | 01:41	05:30 | —
$\|C\| = 409\, Mw$	14:21 | 03:27	11:00 | —

$\hat{T}_{comm}$ includes the per machine times for n-gram cache fetch: local access and remote ($\hat{T}_{RemoteFecth}$). $\hat{T}_{Glue}$ is the local per machine glue calculation time. For the static prefetching cache $\hat{T}_{RemoteFecth}$ is zero. The cache static prefetching time of the nonsingletons is accounted for in phase one, overlapped with counting.

4.4 Cache Alternatives

For glue $g_{2\cdots6}$ and three *corpus* sizes, Table 4 shows the cache miss ratio and size, and the efficiency of the glue calculation (values shown as triples $\{(mr);(Size);(E)\}$) for a single machine. This efficiency (with $K = 1$) reflects the ratio of the communication overheads suffered by a single real machine *versus* an ideal machine, i.e., $E = T_0/T_1$. Miss ratio and size are analyzed first, followed by the efficiency.

Cache Miss Ratio and Size These values result from the model predictions of the numbers of distinct n-grams and singletons (sec. 3). The values for the 8 Mw and 1 Gw *corpora* agree with the empirical data from real English *corpora* [6]. The first line shows miss ratio and size expressions. Remaining lines show: i) For the on-demand cache, its miss ratio (cache system C_1, ..., C_5), from 11.06% in the 8 Mw *corpus* to 2.11% in the 1 Tw *corpus*, and its size (the number of distinct n-grams) – sec. 4.1; ii) For the dynamic cache with Bloom filter, its miss ratio, from 0.76% in the 8 Mw *corpus* to 2.08% in the 1 Tw *corpus* (where the singletons have practically disappeared, Figure 2b), and its size (the number of nonsingletons) – sec. 4.2; iii) For the static prefetching case of the `FAset` filled with all the nonsingletons – sec. 4.3, the miss ratios of 43.3%, 27.7% and 0.04%, respectively, for the 8 Mw, 1 Gw and 1 Tw *corpora*, are due to the singleton misses, not involving any fetching overhead, leading to a miss ratio of 0%.

Table 4: Cache alternatives ($K = 1$, $g_{2\cdots6}$) — Miss ratio, Cache size (number of n-grams in units of $M = 10^6$ or $G = 10^9$), Efficiency

	Dynamic	**Dynamic with Bloom filter**	**Static**
***Corpus* size**	(mr **%);(Size);(E %)**	(mr **%);(Size);(E %)**	(mr **%);(Size);(E %)**
Generic	$\left(\frac{D_{All}}{all_g}\right); (D_{All}); (E_D)$	$\left(\frac{NS_{All}}{all_g}\right); (NS_{All}); (E_{BF})$	$(mr_S); (\lvert FA_{set}\rvert); (E_S)$
Small (8 Mw)	(11.06); (23M); (6)	(0.76); (1.6M); (49)	$\left(0^{\star}\right)$; (1.6M); (100)
Large (1 Gw)	(9.02); (1.8G); (7)	(1.32); (0.27G); (34)	$\left(0^{\star\star}\right)$; (0.27$G$); (100)
Very large (1 Tw)	(2.11); (65G); (20)	(2.08); (64G); (20)	$\left(0^{\star\star\star}\right)$; (64$G$); (100)

$D_{All}=\left|D_{All_{in}C}\right|$; $S_{All}\equiv\left|S_{All_{in}C}\right|$; $all_g\equiv\left|all_{g_{2\cdots6}Ref}\right|$

$\left|FA_{set}\right|=\left|D_{All_{in}C}\right|-\left|S_{All_{in}C}\right|=\left|NS_{All_{in}C}\right| \Longrightarrow \left(43.3\%\rightarrow 0^{\star}\right), \left(27.7\%\rightarrow 0^{\star\star}\right), \left(0.04\%\rightarrow 0^{\star\star\star}\right)$

Efficiency The glue efficiency E, for $K \geq 1$ wrt the glue computation in an ideal (no overheads) sequential machine ($T_0 = \left(\sum_{i=2}^{6} |D_i|\right) \times t_{glue}$, for $g_{2\cdots6}$), is:

$$E = \frac{T_0}{K \times \hat{T}} = \frac{T_0}{K \times \left(\frac{T_0}{K} + \hat{T}_{comm}\right)} = \frac{1}{1 + \frac{1}{G}} = \frac{1}{1 + \frac{all_{g_{2\cdots6}Ref}}{\sum_{i=2}^{6}|D_i|} \times \frac{t_{fetch}}{t_{glue}} \times mr} \tag{13}$$

where t_{glue} is the per n-gram local glue time; $\hat{T}$ is the per machine execution time; $\hat{T}_{comm} = (all_{g_{2\cdots6}Ref}/K) \times t_{fetch} \times mr$ is the per machine n-gram misses communication time; t_{fetch} is the per n-gram remote fetch time; and $G = (T_0/K)/\hat{T}_{comm}$ is the computation-to-communication granularity ratio, which includes: i) the algorithm-dependent term $f_a = \left(\sum_{i=2}^{6} |D_i|\right)/all_{g_{2\cdots6}Ref}$, i.e. the number of glue operations per memory reference, being approximately constant with the *corpus* size, for each glue, e.g., around 0.10 for $g_{2\cdots6}$; ii) the measured implementation-dependent ratio $f_i = \frac{t_{fetch}}{t_{glue}} \approx 20$, staying almost constant wrt the *corpus* size and number of machines used ($1 \leftrightarrow 48$); iii) and $1/mr$. Thus, in LocalMaxs $G = \frac{f_a \times f_i}{mr} \approx \frac{0.10/20}{mr} = \frac{0.005}{mr}$. For example, $E \geq 90\% \implies G \geq 10 \implies mr \leq 0.05\%$, which can only be achieved by a static prefetching cache (sec. 4.3). Indeed Table 4 shows that for the on-demand cache the efficiency values are very low, even with Bloom filters where $E \leq 50\%$ always. In general, other methods, exhibiting higher values of the algorithm-dependent term f_a, will require less demanding (i.e., higher) miss ratio values: e.g., if the f_a term is around 100, then $mr = 50\%$ would be sufficient to ensure $E = 90\%$.

5 Conclusions and Future Work

We found out that for the statistical extraction method LocalMaxs the miss ratio of a dynamic on-demand n-gram cache is lower bounded by the proportion of distinct n-grams in a *corpus*. The proportion of distinct n-grams wrt the total number of cache references, i.e. the miss ratio, decreases monotonically with the *corpus* size tending to an asymptotic *plateau*, e.g., ranging from 11% (for the smaller *corpora*) to 1.5% (in the *plateaux* region) for English *corpora* when considering a single machine. However, these miss ratio values imply very low efficiency of the glue calculation wrt an ideal sequential machine, from 6% to 26%. This is due to the significant amount of cold-start n-gram misses needed to fill up the n-gram cache with the frequencies of all distinct n-grams. To overcome these limitations we have shown that Bloom filters or static prefetching can significantly improve on the cache miss ratio and size. Bloom filters benefits were found related to the distribution of the singletons along the entire *corpus* size spectrum. By extending a previously proposed theoretical model [5], we found out that the number of singletons first increases with the *corpus* size until a maximum and then it decreases gradually, tending to zero as the singletons disappear for very large *corpora*, e.g., in the Tw region for the English *corpora*. This behavior of the singletons determines the effectiveness of the Bloom filters which achieve a reduction of the miss ratio, namely, to the range from 1% (for the smaller *corpora*) to 1.5% (in the *plateaux* region) for English *corpora* for a single machine. However, the corresponding efficiency is still low, always below 49%. Hence, using an n-gram cache with static prefetching of the nonsingletons is of utmost importance for higher efficiency. We have shown that, in a multiphase method like LocalMaxs where it is possible during a 1^{st} phase (in overlap with the n-gram frequency counting), to anticipate and prefetch the set of n-grams needed, then one can ensure a 0% miss ratio in a 2^{nd} phase for glue calculation, leading to 100% efficiency. For a static prefetching cache (sec. 4.3), it is possible, by design, to keep a constant miss ratio, leading to a constant efficiency wrt the number of machines. The above improvements were implemented within the LocalMaxs parallel and distributed architecture (sec. 2), experimentally validated for *corpora* up to 1 Gw. Although this study was conducted in the context of LocalMaxs, the main achievements apply to other statistical multiphase methods accessing large scale n-gram statistical data, thus potentially benefiting from an n-gram cache. For *corpora* beyond 1 Gw we conjecture that the global behavior of the n-gram distribution, as predicted, remains essentially valid, as the model relies on the plausible hypothesis of a finite n-gram vocabulary for each language and n-gram size, at each temporal epoch. We will proceed with this experimentation for *corpora* beyond 1 Gw, although fully uncut huge Tw (10^{12} words) *corpora* are not easily available yet [26].

References

1. Google Ngram Viewer. [Online]. Available: https://books.google.com/ngrams
2. D. Lin et al., "New Tools for Web-Scale n-grams," in *LREC*, 2010.
3. J. F. da Silva et al., "Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units," in *Procs. of the* 9^{th} *Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence*, ser. EPIA '99. Springer Berlin Heidelberg, 1999, pp. 113–132.
4. ——, "A Local Maxima Method and a Fair Dispersion Normalization for Extracting Multiword Units," in *In Procs. of the* 6^{th} *Meeting on the Mathematics of Language*, 1999, pp. 369–381.
5. ——, "A Theoretical Model for n-gram Distribution in Big Data Corpora," in *2016 IEEE International Conference on Big Data*, pp. 134–141.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22741-8_6

6. Parallel LocalMaxs. [Online]. Available: http://cjsg.ddns.net/~cajo/phd/
7. D. Arroyuelo et al., "Distributed Text Search Using Suffix Arrays," *Parallel Computing*, vol. 40, no. 9, pp. 471–495, 2014.
8. C. Goncalves et al., "A Parallel Algorithm for Statistical Multiword Term Extraction from Very Large *Corpora*," in *IEEE* 17*th* *International Conference on High Performance Computing and Communications*, 2015, pp. 219–224.
9. ——, "An *n*-gram Cache for Large-scale Parallel Extraction of Multiword Relevant Expressions with LocalMaxs," in *IEEE* 12*th* *International Conference on e-Science.* IEEE Computer Society, 2016, pp. 120–129.
10. B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications ACM*, vol. 13, no. 7, pp. 422–426, 1970.
11. B. Daille, "Study and Implementation of Combined Techniques for Automatic Extraction of Terminology," in *The Balancing Act: Combining Symbolic and Statistical Approaches to Language.* MIT Press, 1996.
12. P. Velardi et al., "Mining the Web to Create Specialized Glossaries," *Intelligent Systems, IEEE*, vol. 23, no. 5, pp. 18–25, 2008.
13. D. Pearce, "A Comparative Evaluation of Collocation Extraction Techniques," in 3*rd* *International Conference on Language Resources and Evaluation*, 2002.
14. K. W. Church et al., "Word Association Norms, Mutual Information, and Lexicography," *Comput. Linguist.*, vol. 16, pp. 22–29, 1990.
15. T. Dunning, "Accurate Methods for the Statistics of Surprise and Coincidence," *Comput. Linguist.*, vol. 19, pp. 61–74, 1993.
16. K. W. Church et al., "Concordance for Parallel Texts," in 7*th* *Annual Conference for the new OED and Text Research*, 1991, pp. 40–62.
17. C. Goncalves, "Parallel and Distributed Statistical-based Extraction of Relevant Multiwords from Large Corpora," Ph.D. dissertation, FCT / UNL, 2017.
18. G. K. Zipf, "The Psychobiology of Language: An Introduction to Dynamic Philology," in *MIT Press*, 1935.
19. B. B. Mandelbrot, "On the Theory of Word Frequencies and on Related Markovian Models of Discourse," in *Structures of Language and its Mathematical Aspects.* American Mathematical Society, 1961, vol. 12, pp. 134–141.
20. R. Kuhn, "Speech Recognition and the Frequency of Recently Used Words: A Modified Markov Model for Natural Language," in *Procs. of the* 12*th* *Conf. on Computational Linguistics*, ser. COLING '88, vol. 1. ACM, 1988, pp. 348–350.
21. Breslau et al., "Web Caching and Zipf-like Distributions: Evidence and Implications," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, Mar 1999, pp. 126–134 vol.1.
22. Baeza-Yates et all., "The Impact of Caching on Search Engines," in *Procs. of the* 30*th* *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '07. ACM, 2007, pp. 183–190.
23. Q. Yang et al., "Web-log Mining for Predictive Web Caching," *IEEE Trans. on Knowl. and Data Eng.*, vol. 15, no. 4, pp. 1050–1053, Jul. 2003.
24. A. S. Balkir et al., "A Distributed Look-up Architecture for Text Mining Applications Using MapReduce," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011, pp. 1–11.
25. Luna Cloud. [Online]. Available: http://www.lunacloud.com
26. T. Brants et al., "Large Language Models in Machine Translation," in *Procs. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007, pp. 858–867.