A Supplemental Documentation

This paper recommends best practices does not construct a new dataset or benchmark; thus not all parts of the recommended supplemental material (e.g. a model card or datasheet) are applicable. However, since we host an implementation and plan to distribute artifacts and code, we address similar relevant concerns:

- URL, landing page, and demo: dataportraits.org
- Code: to be released through the site above. Open source code will be hosted on GitHub.
- Hosting and Preservation: We will maintain the existing interactive web interfaces for at least one year from publication. Code will be available for long-term distribution through GitHub.

B Sketch Misses

In Figure 2, consider the query string: `defg`. This string lies on the boundary of the n-grams split during corpus processing. No stride 1 width 4 ngrams extracted from that query will match the database. However, if the query string were expanded to length $2 \cdot width - 1$, note that it would necessarily intersect at least one of the hashed n-grams. For example, `defghij` will match at `fghi`. In this way, given a long enough query string, our query protocol guarantees that at least one match will be found if the query string of interest does occur in the corpus.

C Adversarial Matches

We describe a protocol for *chaining* matches together. In Figure 2, the three matches of `bcde`, `fghi`, `jklm` occur separated by $width$ indexes. Therefore we infer the whole string (formed by concatenating the three n-grams) was present in the corpus. However, this might not be true. If an adversary knew the details of the sketch width (and initial offset into the sequence), they could construct a document that embeds n-grams in different locations, such that a query string would appear to be present according to our protocol. For example, the sketch in Figure 2 would falsely infer that the string `fghibcde` is present in the corpus, since it is composed of two chosen matches. This is very unlikely in practice, given appropriately chosen widths and sketch resolutions. This is essentially a permutation attack, and a similar approach could be used to fool a BM-25 index.

D WMT Documents

Table 5 lists the full test set, doc id, and approximate longest match results from Figure 3.

E Counting Expected Matches

Consider a string of interest S, with length N that is embedded in a larger document D. Matching S with a strided Bloom filter with width w will yield a chain of something around $\frac{N}{w}$ tiles (substrings of length w). For example, take a sketch with width $w = 50$ and a string with $N = 150$. If the tiles in D are perfectly aligned on the boundaries of S, the Bloom filter will find $150/50 = 3$ matches. Perfectly aligned means that string S begins at an offset in D that is a multiple of w — so when breaking D into non-overlapping chunks (tiles) of size w, the start of some tile is also the start of S.

This perfect alignment might happen by chance, but most likely there will be some parts of S that hang over the tile boundaries, meaning only some inner part of S will match the hashed tiles. In Figure 2 the query string $S = abcdefghijklmn$ is not perfectly aligned. *a* and *n* hang over and thus the only complete tiles are the inner *bcde*, *fghi*, *jklm* tiles match.

Given the width and length of a string, we can calculate the expected number of matches for any possible alignment of the string. Note that there are w possible alignments and each is equally likely.

A string of length N modulo the tile width w can be written as $N = aw + b$ where a is the number of full tiles and b is the remainder. Consider alignments other than the perfect one boundary. We have

Table 5: Full WMT overlap information.

Test Set	doc_id	Longest
wmt13.en-fr.ref	lemondefr/2012/12/01/275696	700
wmt16.en-fi.ref	kaleva.fi.29723	400
wmt16.en-de.ref	tagesspiegel.de.65447	350
wmt20.en-iu.ref	nunatsiaq-20190930	300
wmt12.en-de.ref	noroeste/2011/11/15/78596.html	250
wmt16.en-de.ref	borkenerzeitung.de.56604	250
wmt14.en-fr.ref	4bb85eb6281e0b19986de1d4f867e3ff	250
wmt15.en-ru.ref	893-kommersant	200
wmt18.en-fi.ref	karjalainen.fi.65284	200
wmt18.en-ru.ref	kommersant.324314	200
wmt14.en-fr.ref	cd085bbb218a7afc1255b2b60a06692a	200
wmt15.en-de.ref	14428-abendzeitung-muenchen.de	200
wmt15.en-ru.ref	115-aif	200
wmt16.en-ru.ref	lgng.30237	150
wmt16.en-ro.ref	ziare.ro.17378	150
wmt15.en-ru.ref	1375-rg.ru	150
wmt14.en-fr.ref	90c566f54bf1076e6f539875d45d673c	150
wmt17.en-ru.ref	izvestiya.51251	150
wmt16.en-ro.ref	hotnews.ro.8884	150
wmt14.en-fr.ref	96e21a07ed57d79665a35a548ef7d841	150
wmt16.en-de.ref	abendzeitung-nuernberg.de.12297	150
wmt17.en-de.ref	dw.47065	150
wmt18.en-de.ref	handelsblatt.com.180784	150
wmt17.en-de.ref	frankfurter-rundschau.70094	150
wmt13.en-fr.ref	cyberpresse/2012/12/01/1564248	100

$b+1$ alignments that produce a matching tiles. We will also have $w-b-1$ alignments that produce $a-1$ tiles. Summing and cancelling terms, we have $(b+1)a+(w-b-1)(a-1) = aw-w+b+1$ possible matching tiles. Substituting the length of the string simplifies to $N-w+1$ possible matches, and since each w alignment is equally likely, the expected number of matches is $E(N,w) = \frac{N-w+1}{w}$. The 4 possible alignments with 11 possible matching strings in the Figure 2 example are:

```
[abcd, efgh, ijkl] (missing mn)
[bcde, fghi, jklm] (missing a, n)
[cdef, ghij, klmn] (missing ab)
[defg, hijk] (missing abc, lmn)
```