|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| curious that the p[ob]=.1, blue line, is on the whole the most optimal choice here. | | | | | | | |  |  |
| so, on average, one out of every 10 query terms will be an obfuscation term. This does | | | | | | | | |  |
| better than, say, 3 out of 10 or 9 out of 10. it may be the case that the obfuscations have | | | | | | | | |  |
| a different distribution than the real terms (I just used a uniform for obfuscations, but a | | | | | | | | |  |
| zipf for real terms -- making obfuscations sort of match the real distribution may prove especially | | | | | | | | | |
| effective -- just some speculation, I haven't found any actual evidence for this yet. | | | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | it's interesting to note that the real terms consist of a dictionary of size 50, | | | | | | | |
|  | but obfuscations are effective at mitigating the maximum likelihood attack. | | | | | | | |
|  |  |  |  |  |  |  |  |  |
|  | this is for many reasons: | | |  |  |  |  |  |
|  | (1) curse of dimensionality -- as parameters increase, size of space | | | | | | |  |
|  | you must explore increases exponentially (n!). So, given n real | | | | | | |  |
|  | terms and k obfuscation terms, we have (n+k)! possibilities. This | | | | | | |  |
|  | is (n+k)\*(n+k-1)\*…\*(n+1) times larger than the system without | | | | | | |  |
|  | obfuscations. | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | combining this with secrets is even better (at the cost of increased index | | | | | | | |
|  | size, build time, and load time): if each real term has m secrets, | | | | | | |  |
|  | then we have (nm)! Possibilities. This is a huge increase. And, if we | | | | | | |  |
|  | also add obfuscations to this, then it's (nm+k)! | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | (2) you must see more samples (larger history) for the same confidence | | | | | | |  |
|  | level. For instance, if real term t appears once every k times, but then | | | | | | |  |
|  | we add m secrets, then hidden terms associated with t appear only | | | | | | |  |
|  | k/m times, which means you need to see more before the law of large | | | | | | |  |
|  | numbers takes over. | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NOTE: psip does better likely because, in psip, the noisy freq for | | | | | | |  |  |
| a term is (true\_freq) + uniform(0, percent\_error \* true\_freq), where | | | | | | |  |  |
| as for psif it is more correctly (true\_freq) +- uniform(0, percent\_error \* true\_freq) | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |
| otherwise, I predict they would be the same. | | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| NOTE #2: bsib and psip have a built-in frequency error based on their | | | | | | |  |  |
| blocks, where terms (unigrams/bigrams) are either a member of the | | | | | | |  |  |
| block or not a member (set membership). No multiplicities stored | | | | | | |  |  |
| per block, although doing multiplicities would be possible. | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | compared to precision, bm25 map is less sensitive to fp rate. this makes sense, because | | | | | | | | |  |
|  | if a false positive happens when measuring precision, it will admit a term that should | | | | | | | | |  |
|  | not be included in the result set which will effect precision. | | | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | however, with bm25 map, we are ranking the documents. So, even if a document is | | | | | | | | |  |
|  | falsely hitting on a search term, it’s the order that counts. | | | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | curious: what about this -- take N documents, the probability of a false positive on | | | | | | | |  |  |
|  | the corpus is N\*(fp\_rate). If N is really large, this could have interesting behavior | | | | | | | |  |  |
|  | with the term weighting. Need to collect thoughts on this still. | | | | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | | | | | | | | | |  | |  | |  | |  | |
|  | |  | | | | | | | | | |  | |  | |  | |  | |
|  | |  | | | |  | |  | |  | |  | |  | |  | |  | |
|  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  | |  | | | | | | | | | |  | |  | |  | |  | |
|  | |  | | | | | | | | | | | |  | |  | |  | |
|  | |  | | | | | | | |  | |  | |  | |  | |  | |
|  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  | |  | | | | | |  | |  | |  | |  | |  | |  | |
|  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  | |  | |  | | | | | | | | | | | |  | |  | |
|  | |  | |  | | | | | | | | | | | |  | |  | |
|  | | |  |  |  |  | | --- | --- | --- | --- | | Psib is using a data structure that is designed to be | | | | | efficient for small to medium sized documents. If | | | | | documents are very large, it is advisable to break them | | | | | up into multiple indexes -- this is a standard practice | | | | | in IR. |  |  |  | |  |  |  |  | | It's a trade-off. | |  |  | |  |  |  |  | | |  | | | | | | | | | |  | |  | |  | |
|  | |  | |  | | | | | | | | | | | |  | |  | |
|  | |  | |  | | | | | |  | |  | |  | |  | |  | |
|  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  | |  | |  | | | | | | | | | |  | |  | |  | |
|  | |  | |  | | | | | | | | | |  | |  | |  | |
|  |  | | | | | | | |  | |  | |  | |  | |  | |  | |
|  |  | |  | |  | |  | |  | | | |  | |  | |  | |  | |
|  |  | |  | |  | |  | |  | | | | | | | |  | |  | |
|  |  | |  | |  | |  | |  | | | | | | | |  | |  | |
|  |  | |  | |  | |  | |  | | | | | | | |  | |  | |
|  |  | |  | |  | |  | |  | | | | | | | | | | | |
|  |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  |  | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | |  | |
|  |  | | | | | | | | | | | |  | |
|  |  | | | | | |  | |  | |  | |  | |
|  |  | |  | |  | |  | |  | |  | |  | |
|  |  | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | |
|  |  | | | | | | | | | | | | | |
|  |  | | | |  | |  | |  | |  | |  | |
|  |  | |  | |  | |  | |  | |  | |  | |
|  |  | |  | |  | |  | |  | |  | |  | |