6.889 Final Project: Comparing Monthly Distributions of the Most Common Phrases on the Internet in 2008-2009

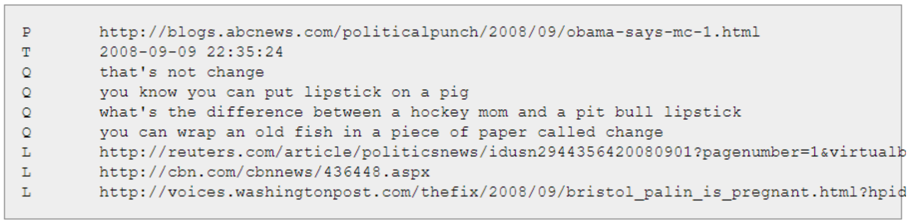
Isabella Kang, Xinyu Lin, Alice Zhang

December 10, 2020

1 Introduction

Over time many distinctive turns of phrase become popular and then eventually fall away from use. How does the frequency of popular phrases change from month to month? To investigate this question, we obtained data regarding the use of common phrases in online news sources and blog posts. Then, using both exact and sublinear methods, we obtain empirical results about the differences in the distribution of phrase use across different months, as well as the accuracy of the sublinear algorithms.

The data we used is from the Memetracker project, which analyzed around 900,000 news stories and blog posts per day over the course of nine months, from August 2008 to April 2009. All of it is available, grouped by month, on the SNAP website. Figure 1 gives an example of the Memetracker data for a single webpage.



```
P       http://blogs.abcnews.com/politicalpunch/2008/09/obama-says-mc-1.html
T       2008-09-09 22:35:24
Q       that's not change
Q       you know you can put lipstick on a pig
Q       what's the difference between a hockey mom and a pit bull lipstick
Q       you can wrap an old fish in a piece of paper called change
L       http://reuters.com/article/politicsnews/idusn2944356420080901?pagenumber=1&virtualb
L       http://cbn.com/cbnnews/436448.aspx
L       http://voices.washingtonpost.com/thefix/2008/09/bristol_palin_is_pregnant.html?hpid
```

Figure 1: An example of the Memetracker data for a single webpage.

The top line, labeled "P," indicates the URL where the quotes were found. The next line, labeled "T," is the timestamp. The lines labeled "Q" contain the common phrases identified on the page, and the lines labeled "L" are the hyperlinks found on the page. Our analysis required only the common phrases. Another interesting possibility to consider would have been to use the URLs as well to represent the webpages as a graph, and to test graph properties, but for this analysis we restricted ourselves to testing the distributions of the phrases.

Our two sublinear tests, the code for which can be found here, are motivated by our interest in how internet memes spread: how long do memes persist through time? Do memes quickly fade from popularity after a month? The first type of analysis we did to we answer these questions was to estimate the ℓ_2 distance between the distributions of the phrases across consecutive months.

We sampled the data to estimate the ℓ_2 distance in sublinear time, but we also calculated the ℓ_2 distance exactly using brute force so that we could compare the exact results with our estimations. The second algorithm we ran was to test the closeness in ℓ_1 norm of the distributions over phrases, again in sublinear time.

2 Testing Closeness of Two Distributions With Access to Samples

The first idea we wanted to explore was testing if the phrase distributions were "similar" between consecutive months, and how this similarity changed throughout time. We followed an algorithm by Chan et. al [3] and also present a brief overview of their proof below. This algorithm estimates the ℓ_2 distance between two distributions for which we only have sample access.

Let P and Q be the distributions we have sample access to. Then from our samples, we can calculate a statistic $\mathcal{Z}$ satisfying the following inequality:

$$\Pr\left(\left|\frac{\mathcal{Z}}{m^2} - ||P-Q||_2^2\right| \geq \epsilon^2\right) \leq \frac{1}{10}$$

We would like to compute the estimation of $||P-Q||_2^2$, which is $\frac{\mathcal{Z}}{m^2}$, and we will calculate it using the following steps:

Algorithm 1 : ℓ_2 Distance Estimator

```
1: Draw $Poisson(m)$ samples from both $P$ and $Q$. (We use Single-Poissonization to achieve independence
   between samples.)
2: For each element $i$, we count the number of instances of that element from each of the distributions $P$
   and $Q$. Let these values be $X_i$ and $Y_i$, respectively.
3: Calculate $\mathcal{Z} = \sum_{i=1}^{n}(X_i - Y_i)^2 - X_i - Y_i$.
4: Output $\frac{\mathcal{Z}}{m^2}$
```

Proof. We will calculate the expectation and variance of the statistic and use Chebyshev's Inequality to prove the inequality above.

We will first show that $E(\mathcal{Z}) = m^2||P-Q||_2^2$. Note that after Single-Poissonization, we have that $X_i \sim Poisson(mP_i)$ and $Y_i \sim Poisson(mQ_i)$. We have that

$$\begin{aligned}
E(\mathcal{Z}) &= \sum_{i=1}^{n} E[(X_i - Y_i)^2 - X_i - Y_i] \\
&= \sum_{i=1}^{n} E(X_i^2 - X_i) - 2E(X_i)E(Y_i) + E(Y_i^2 - Y_i) \\
&= \sum_{i=1}^{n} m^2P_i^2 - 2m^2P_iQ_i + m^2Q_i^2 \\
&= \sum_{i=1}^{n} m^2(P_i - Q_i)^2 = m^2||P_i - Q_i||_2^2
\end{aligned}$$

which means that $E(\frac{\mathcal{Z}}{m^2})$ is exactly the ℓ_2 distance.

Next we will show that $Var(\mathcal{Z}) \leq 8m^3||P-Q||_4^2\sqrt{b} + 8m^2b$, where $b = \max(||P||_2^2, ||Q||_2^2)$. By independence, we have that

$$\begin{aligned} Var(\mathcal{Z}) &= \sum_{i=1}^{n} Var((X_i - Y_i)^2 - X_i - Y_i) \\ &= \sum_{i=1}^{n} E[((X_i - Y_i)^2 - X_i - Y_i)^2] - (E[(X_i - Y_i)^2 - X_i - Y_i])^2 \end{aligned}$$

Rewriting the expression above using the formulas for the second moments of the Poisson distribution and using Cauchy-Schwartz, we get

$$\begin{aligned} \sum_{i=1}^{n} 4m^3(P_i - Q_i)^2(P_i + Q_i) + 2m^2(P_i + Q_i)^2 &\leq \sum_{i=1}^{n} 4m^3 \sqrt{\sum_{i=1}^{n}(P_i - Q_i)^4 \sum_{i}(P_i + Q_i)^2} + 2m^2(P_i + Q_i)^2 \\ &\leq 8m^3||P - Q||_4^2\sqrt{b} + 8m^2 b \end{aligned}$$

where $b = \max(||P||_2^2, ||Q||_2^2)$, as before.

Finally, putting everything together with Chebyshev's Inequality, we have that

$$\begin{aligned} \Pr\left(\left|\frac{\mathcal{Z}}{m^2} - ||P - Q||_2^2\right| \geq \epsilon^2\right) &= \Pr\left[|\mathcal{Z} - E(X)| \geq m^2\epsilon^2\right] \\ &\leq \frac{Var(\mathcal{Z})}{m^4\epsilon^4} \leq \frac{1}{10} \end{aligned}$$

where the last inequality follows when we choose m such that $m \geq \Theta\left(\frac{\sqrt{b}||P-Q||_4^2}{\epsilon^4} + \frac{\sqrt{b}}{\epsilon^2}\right)$

$\square$

2.1 Experimental Results and Performance

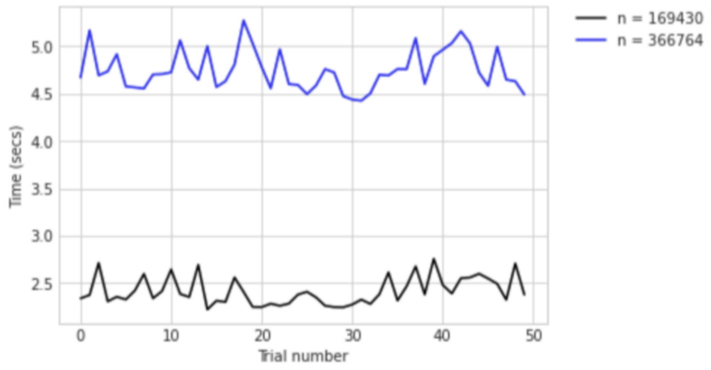


Figure 2: Runtime over 50 trials calculating ℓ_2 distance by brute force. The black and blue lines indicate the time when running the algorithm between Nov-Dec 2008 and Feb-Mar 2009, respectively.

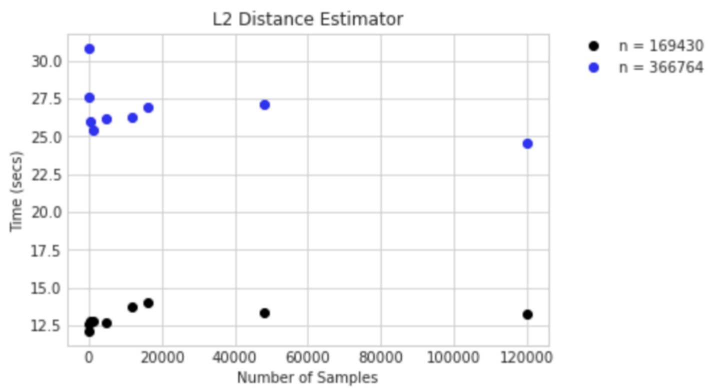


Figure 3: The runtime of running algorithm 1 on various sample sizes. The black and blue dots indicate the distribution of phrases over 2008-11 vs. 2008-12 and 2009-02 vs. 2009-03 respectively. The sample sizes are $\{120, 160, 480, 1200, 4800, 12000, 16000, 48000, 120000\}$.

It's interesting to note in the graph above that increasing the number of samples by more than a factor of 5 did not cause a marked difference in the runtime (in fact, the graph even suggests that the time decreased, which is attributed to noise). The runtime graphs in Section 3 show results that are more of what we would expect - a linear relationship between runtime and number of samples.

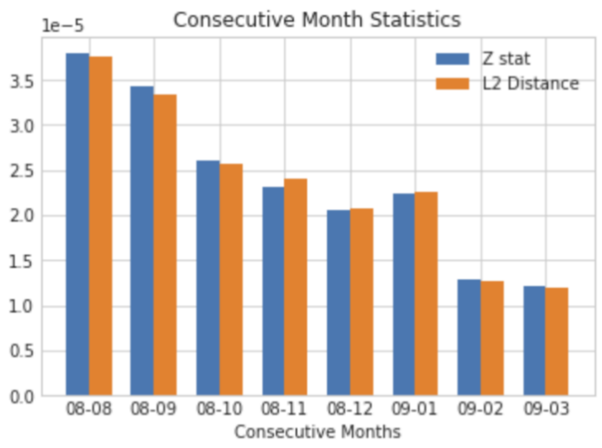


Figure 4: The comparison between the ℓ_2 distance and the $\mathcal{Z}$ statistic when running algorithm 1 for consecutive months, where e.g. the label $08 - 09$ represents Year 2008 and the ℓ_2 difference between month 9 (September) and month 10 (October).

This graph above is where we explore the month-to-month ℓ_2 distances, all compared against each other. We can see a very clear downward trend in the ℓ_2 distance when we look at the metric across consecutive months. Intuitively, this means that memes were changing very rapidly in August 2008, but much more slowly as 2008 ended and 2009 began. To figure out why, we can take a look at a few of the most common memes in August and September 2008, where the counts are listed to the left:

```
August 2008

31287 : san array model reference
22948 : gang of ten
19328 : my work queue
11149 : the dark knight
10543 : intel server support
10413 : ibm server support
```

```
September 2008

27568 : alle kids sind vips
25763 : gang of ten
23662 : my work queue
12580 : bridge to nowhere
10985 : san array model reference
10097 : like a virgin
```

We can see that although there are a few overlaps, many of the phrases are related to briefly exciting events (such as movies or political proposals) occurring in a certain month. Furthermore, the 2008 Summer Olympics in Beijing had also just started in August 2008, so some memes related to that event may have risen in temporary popularity that month (though oddly enough, none of the top memes in August or September seem to have any relation to the Olympics). We can compare the top memes in these two volatile months with the top memes in the months with the smallest ℓ_2 distance below:

```
March 2009

52846 : the high priest of deregulation
36529 : phil gramm is the single most important reason for the current financial crisis
34602 : c tait m gara faubourg de carthage
31571 : var headline escape
23422 : my work queue
21766 : needs to be defeated
17104 : abandon all hope
17096 : that is equal to the task ahead
16962 : six months in the
16721 : community spotlight submission
```

```
April 2009

29402 : the high priest of deregulation
26728 : my work queue
20318 : phil gramm is the single most important reason for the current financial crisis
19287 : var headline escape
16391 : dancing with the stars
15264 : c tait m gara faubourg de carthage
12103 : needs to be defeated
10917 : report inappropriate comment
10596 : war on terror
9522 : abandon all hope
```

We can see here that many of the top memes generally persisted across months, and represent long-term economic trends that people may have been worried about following the 2008 elections, so it's logical that the ℓ_2 distance was much smaller (by more than a factor of 2).

If we had more data on several more months, perhaps we could explore whether seasonality had anything to do with the trend we saw above, since it's possible that winter months result in less activity occurring overall, leading to a decrease in the breadth of rapidly changing online memes.

We also investigate the relationship between number of samples and the estimated ℓ_2 distance error. Figure 5 illustrates the difference between the estimated and expected ℓ_2 distance for different number of samples for 2009-02 and 2009-03. After taking log scale on the number of samples, the plot is still non-linear. This implies that increasing the number of samples linearly has a negative super-linear effect on the estimated error. However, there is a small spike at the beginning of the plot, which may have been due to some sampling noise since we were using a small sample size.

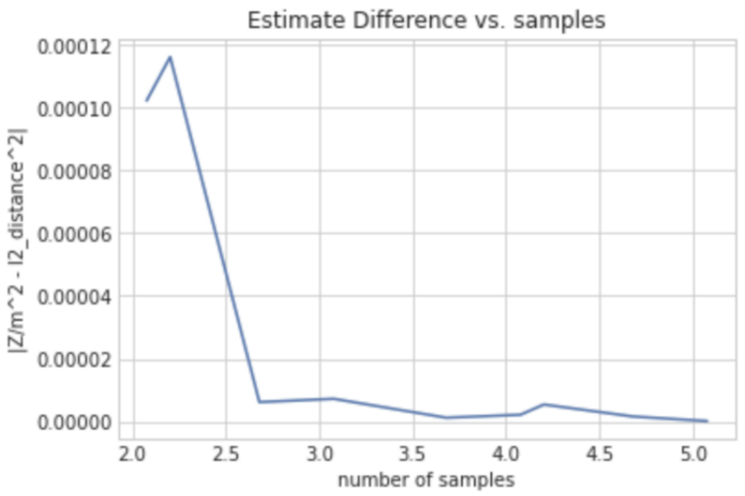


Figure 5: Error $|\frac{\mathcal{Z}}{m^2} - L2^2|$ vs log number of samples.

3 Testing Closeness of Distributions

In this section, we survey another closeness tester introduced in [1]. Once again, we want to test whether two distributions p and q are close in ℓ_1 norm with sublinear query complexity in the size of the domain of the distributions. Like the tester from the previous section, we are given access to both distributions via black boxes. The main theorem is:

Theorem 1. *Given parameters δ and ϵ, and distributions p and q over a set of n elements, there is a test which runs in $O(n^{2/3}\epsilon^{-8/3}log(\frac{n}{\delta}))$ time such that, if $||p-q||_1 \leq max(\frac{\epsilon^{4/3}}{32n^{1/3}}, \frac{\epsilon}{4\sqrt{n}})$, then the test accepts with probability at least $1-\delta$, and if $||p-q||_1 > \epsilon$, then the test rejects with probability at least $1-\delta$.*

In order to prove this theorem, we give a test that determines if distributions p and q are close in ℓ_2 norm. This test is based on estimating self-collision and collision probabilities of p and q. If the two distributions are close, we can expect each self-collision probability to be close to the pairwise collision probability. We provide another theorem that captures this intuition:

Theorem 2. *Given parameters δ and ϵ, and distributions p and q over a set of n elements, there is a test which runs in $O(\epsilon^{-4}log(\frac{1}{\delta}))$ time such that, if $||p-q||_2 \leq \frac{\epsilon}{2}$, then the test accepts with probability at least $1-\delta$, and if $||p-q||_2 > \epsilon$, then the test rejects with probability at least $1-\delta$.*

The test used to prove Theorem 2 is shown in Algorithm 2. We use m to represent the number of samples needed by the test to get constant confidence. It suffices to set $m = O(\frac{1}{\epsilon^4})$ to bound the ℓ_2 distance between p and q by ϵ.

Since $||v||_1 \leq \sqrt{n}||v||_2$, we can extend the ℓ_2-Distance-Test to a ℓ_1-Distance Test by setting $\epsilon' = \frac{\epsilon}{\sqrt{n}}$. This would give the correct output behavior of the test, but due to the order of dependence on ϵ in the ℓ_2-Distance-Test, the resulting runtime will be quadratic in n. However, it is possible to achieve sublinear runtime if the input distributios are reasonably evenly distributed. We formalize this in lemma 4 by analyzing the dependence of the variance of s and r on the parameter $b = max(||p||_\infty, ||q||_\infty)$. In particular, we will show when $b = O(n^{-\alpha})$, we can call ℓ_2-Distance-Test with an error parameter of $\frac{\epsilon}{\sqrt{n}}$ with a runtime of $O(\epsilon^{-4}(n^{1-\alpha/2} + n^{2-2\alpha}))$.

Recall that one naive tester for closeness testing is to get sufficiently many samples from p and q in order to learn each distribution to accuracy $O(\epsilon)$. Then we check the closeness of the estimated distributions. When the minimum probability element is quite large, this naive algorithm can be more efficient. Thus, we also suggest a filtering algorithm that separates the domain of the distribution being tested into two parts: (1) the big elements to which the distributions assign relatively high probability weight and (2) the small elements, which are the other elements. In particular, we use definition 3 to identify elements with large weights. After filtering out the big elements, we apply the naive tester to the distribution restricted to big elements, and the tester based on ℓ_2 distance to distribution restricted to small elements.

Definition 3. *An element i is called big with respect to a distribution p if $p_i > (\frac{\epsilon}{n})^{2/3}$.*

The complete ℓ_2 test is given below in algorithm 2. This test is used to prove theorem 1.

Algorithm 2 : ℓ_2-Distance-Test(p, q, m, ϵ, δ)

```
1: Repeat $O(log(\frac{1}{\delta})$ times:
2:     Let $F_p$ and $F_q$ be $m$ samples from $p$ and $q$ respectively
3:     Let $r_p$ and $r_q$ number of pairwise self-collisions in $F_p$ and $F_q$ respectively
4:     Let $Q_p$ and $Q_q$ be $m$ samples from $p$ and $q$ respectively
5:     $s_{pq} \leftarrow$ number of collisions between $Q_p$ and $Q_q$
6:     $r \leftarrow \frac{2m}{m-1}(r_p + r_1)$
7:     $s = 2s_{pq}$
8:     If $r - s > \frac{3}{4}m^2\epsilon^2$, reject current iteration
9: Reject if majority of iterations reject, accept otherwise
```

Algorithm 3 : ℓ_1-Distance-Test(p, q, ϵ, δ)

```
1: $b \leftarrow (\frac{\epsilon}{n})^{2/3}$
2: Let $S_p$ and $S_q$ be $m = O(\epsilon^{-8/3}n^{2/3}log(\frac{n}{\delta}))$ samples from $p$ and $q$ respectively
3: Discard elements in $S_p$ and $S_q$ that occur less than $mb(1 - \frac{\epsilon}{26})$ times
4: If $S_p$ and $S_q$ both empty:
5:     $\ell_2$-Distance-Test($p$, $q$, $O(\frac{n^{2/3}}{\epsilon^{8/3}})$, $\frac{\epsilon}{2\sqrt{n}}$, $\frac{\delta}{2}$)
6: else:
7:     Let $x_i^p$ and $x_i^q$ be the number of times element $i$ appears in $S_p$ and $S_q$ respectively
8:     Reject if $\sum_{i \in S_p \bigcup S_q} |x_i^p - x_i^q| > \frac{\epsilon m}{8}$
9:     Let $p'$ be defined as follows: sample an element from $p$. If the sample is not in $S_p \bigcup S_q$, keep;
10:        otherwise replace with an randomly sampled element from domain. Define $q'$ similarly.
11:    $\ell_2$-Distance-Test($p'$, $q'$, $O(\frac{n^{2/3}}{\epsilon^{8/3}})$, $\frac{\epsilon}{2\sqrt{n}}$, $\frac{\delta}{2}$)
```

3.1 Closeness in ℓ_2 norm

Here, we analyze the ℓ_2-Distance-Test and provide a proof for Theorem 2. r_p, r_q, and s from the tester are estimators for the self-collision probability of p, q, and the collision probability between p and q respectively. If p and q are close, we can expect their self-collision probabilities to be close to the pairwise collision probability. These probabilities are the inner products of these vectors. In particular, if the set F_p of samples from p is given by $F_p^1, \dots, F_p^m$, then for any pair $i, j \in [m], I \neq j$, we have $Pr[F_p^i = F_q^j] = p \cdot p = ||p||_2^2$. Combining these statistics, we can show that $r - s$ is an estimator for $||p - q||_2^2$.

In order to bound the number of samples to estimate $r - s$ with high accuracy, we must first bound the variance of variables s and r. Note that for self-collisions, we only consider samples where $i \neq j$, but this is not the case for pairwise collisions between p and q. To accommodate this, we scale r_p and r_q appropriately. From the previous paragraph, after scaling, we have:

$$E[s] = 2m^2(p \cdot q)$$
$$E[r - s] = m^2(||p||_2^2 + ||q||_2^2 - 2(p \cdot q)) = m^2(||p - q||_2^2)$$

We now introduce a lemma:

Lemma 4. *There is a constant c such that satisfies the following:*

$$Var(r_p) \leq m^2||p||_2^2 + m^3||p||_2^3 \leq c(m^3b^2 + m^2b)$$
$$Var(r_q) \leq m^2||p||_2^2 + m^3||p||_2^3 \leq c(m^3b^2 + m^2b)$$
$$Var(s) \leq c(m^3b^2 + m^2b)$$

where $b = max(||p||_\infty, ||q||_\infty)$

We skip the proof of this lemma for simplicity. If interested, please refer to section 2.1 of [1]. Using this lemma, we introduce a corollary:

Corollary 5. *There is a constant c such that $Var(r - s) \leq c(m^3b^2 + m^2b)$ where $b = max(||p||_\infty, ||q||_\infty)$.*

Proof. Since variance is additive for independent random variables, we can simply apply Lemma 4. ☐

Now, using Chebyshev's inequality, it follows that if we set $m = O(\epsilon^{-4})$, we can achieve an error probability less than $\frac{1}{3}$. With $O(log\frac{1}{\delta})$ iterations, we can achieve an error probability of at most δ. Now, we can analyze the correctness of the algorithm:

Theorem 6. *For two distributions p and q such that $b = max(||p||_\infty, ||q||_\infty)$ and $m = \Omega(\frac{b^2+\epsilon^2\sqrt{b}}{\epsilon^4})$, if $||p - q||_2 \leq \frac{\epsilon}{2}$, then the ℓ_2-Distance-Test accepts with probability at least $1 - \delta$. If $||p - q||_2 > \epsilon$, then the ℓ_2-Distance-Test accepts with probability less than δ. The running overall sampling complexity and runtime are $O(mlog(\frac{1}{\delta}))$.*

Proof. Let $A = r - s$. By Chebyshev's inequality and Corollary 5, we can say for some constant c:

$$Pr[|A - E[A]| > \rho] \leq \frac{c(m^3b^2 + m^2b)}{\rho}$$

Recall that $E[A] = E[r - s] = m^2(||p - q||_2^2)$. Note that the ℓ_2-Distance-Test can distinguish between the $||p - q||_2 \leq \frac{\epsilon}{2}$ vs. $||p - q||_2 > \epsilon$ if A is within $\frac{m^2\epsilon^2}{4}$. We can then bound the error probability:

$$Pr[|A - E[A]| > \frac{m^2\epsilon^2}{4}] \leq \frac{16(m^3b^2 + m^2b)}{m^4\epsilon^4}$$

Therefore, for $m = \Omega(\frac{b^2+\epsilon^2\sqrt{b}}{\epsilon^4})$, the probability is bounded by a constant. This error probability can be reduced to δ by repeating $O(log(\frac{1}{\delta}))$ times. ☐

3.2 Closeness in ℓ_1 norm

The ℓ_1-Distance-Test consists of two parts. In the first part, we determine the big elements (Definition 3) and estimate their contribution to $||p-q||_1$. The second part filters out the big elements and runs ℓ_2-Distance-Test on the filter distribution with error parameter $\frac{\epsilon}{2\sqrt{n}}$. With these substitutions, the number of samples m is $O(\epsilon^{-8/3}n^{2/3})$. We need to show that by sampling $O(\epsilon^{-8/3}n^{2/3}log(\frac{n}{\delta}))$ times, we can estimate the weights of each big elements to have a multiplicative error of $1+O(\epsilon)$, with probability at least $1-\frac{\delta}{2}$.

We first introduce a new lemma:

Lemma 7. *Let $b=\epsilon^{2/3}n^{-2/3}$. In ℓ_1-Distance-Test, after taking $M=O(\frac{n^{2/3}log(\frac{n}{\delta})}{\epsilon^{8/3}})$ samples from p, we define $\hat{p}_i=\frac{l_i^p}{M}$. Then, with probability at least $1-\delta/2$, the following holds: (1) if $p_i\geq(1-\frac{\epsilon}{13})b$, then $|\hat{p}_i-p_i|<\frac{\epsilon}{26}max(p_i,b)$ and (2) if $p_i<(1-\frac{\epsilon}{13})b$, then $\hat{p}_i<(1-\frac{\epsilon}{26})b$.*

Proof. In each case, we use Chernoff bounds to show that for each i:
If $p_i>b$, then

$$Pr[|\hat{p}_i-p_i|>\frac{\epsilon p_i}{26}]<exp(-O(\epsilon^2Mp_i))<exp(-O(\epsilon^2Mb))\leq\frac{\delta}{2n}$$

If $p_i\leq b$, then

$$Pr[|\hat{p}_i-p_i|>\frac{\epsilon p_i}{26}]<Pr[|\hat{p}_i-p_i|>\frac{\epsilon bp_i}{26p_i}]<exp(-O(\frac{\epsilon^2b^2M}{p_i}))<exp(-O(\epsilon^2Mb))\leq\frac{\delta}{2n}$$

The lemma follows by union bound. $\square$

We can now prove the main theorem:

Theorem 8. *For $\epsilon\geq\frac{1}{\sqrt{n}}$, ℓ_1-Distance-Test accepts distributions p and q if $||p-q||_1\leq max(\frac{\epsilon^{4/3}}{32n^{1/3}},\frac{\epsilon}{4\sqrt{n}})$ and rejects if $||p-q||_1>\epsilon$ with probability at least $1-\delta$. The runtime of the tester is $O(\epsilon^{-8/3}n^{2/3}log(\frac{n}{\delta}))$.*

Proof. Let $S=S^p\bigcup S^q$. By assumption, all the big elements of both p and q are in S, and no element has weight less than $(1-\frac{\epsilon}{13})b$. Let $\Delta_1=\sum_{i\in S}|p_i-q_i|$ and $\Delta_2=||p'-q'||_1$. Note that $\Delta_1\leq||p-q||_1$. Thus, $\Delta_2\leq||p-q||_1$ and $||p-q||_1\leq2\Delta_1+\Delta_2$.

Next, we show that the brute-force estimate of Δ_1 is within an additive error of $\frac{\epsilon}{9}$. By lemma 7, the error term on the ith term of the sum is bounded by

$$\frac{\epsilon}{26}(max(p_i,b)+max(q_i,b))\leq\frac{\epsilon}{26}(p_i+q_i+\frac{2\epsilon b}{13}),$$

since p_i and q_i are at least $(1-\frac{\epsilon}{13})b$. The sum over i of these error terms is at most $\frac{2}{(1-\epsilon/13)b}$ elements in S. Also note that $\epsilon\leq2$. Thus, the total additive error is bounded by

$$\sum_{i\in S}\frac{\epsilon}{26}(p_i+q_i+\frac{2\epsilon b}{13})\leq\frac{\epsilon}{26}(2+\frac{4\epsilon}{13-\epsilon})\leq\frac{\epsilon}{9}$$

Note that $max(||p'||_\infty,||q'||_\infty)\leq b+\frac{1}{n}\leq2b$ for any $\epsilon\geq\frac{1}{\sqrt{n}}$. Thus, we can use the ℓ_2-Distance-Test on p' and q' with $m=O(\epsilon^{-8/3}n^{2/3})$ as shown in theorem 2.

If $||p-q||_1<\frac{\epsilon^{4/3}}{32n^13}$, so are Δ_1 and Δ_2. Thus, the first part of the algorithm accepts. Using the fact that $||v||_2^2\leq||v||_1\cdot||v||_\infty$ for any vector v, we get $||p'-q'||_2\leq\frac{\epsilon}{4\sqrt{n}}$. Thus, the ℓ_2-Distance-Test accepts with probability at least $1-\frac{\delta}{2}$. Likewise, if $||p-q||_1>\epsilon$, either $\Delta_1>\frac{\epsilon}{4}$ or $\Delta_2>\frac{\epsilon}{2}$. In this case, either the first part of the algorithm or the ℓ_2-Distance-Test will reject.

The runtime for the first part and the ℓ_2-Distance-Test are $O(n^{2/3}\epsilon^{-8/3}log(\frac{n}{\delta}))$ and $O(n^{2/3}\epsilon^{-8/3}log(\frac{1}{\delta}))$ respectively. The algorithm makes an error when it estimates Δ_1 poorly or when ℓ_2-Distance-Test makes an error. Thus, the error probability is bounded by δ. □

3.3 Experimental Results and Performance

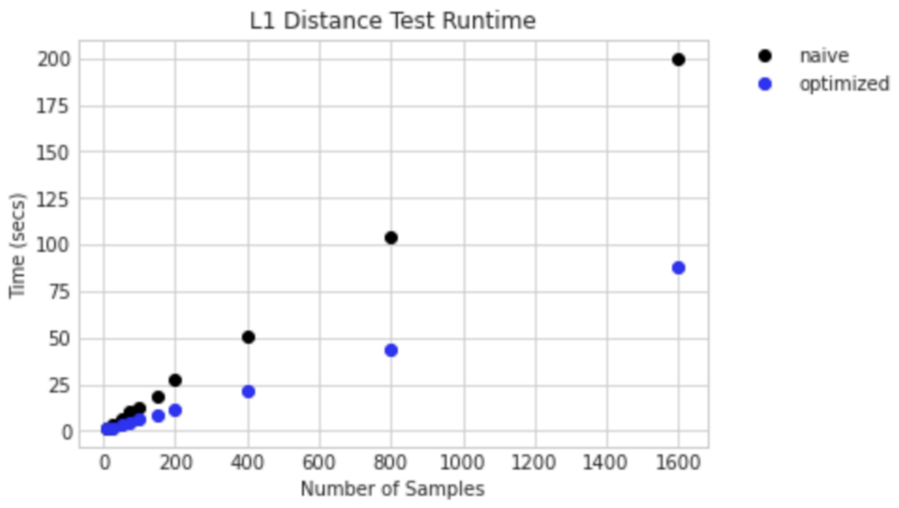


Figure 6: The runtime of running algorithm 3 on various sample sizes. The black and blue dots indicate the distribution of phrases over 2008-11 vs. 2008-12 and 2009-02 vs. 2009-03 respectively. The sample sizes are $\{10, 25, 50, 75, 100, 150, 200, 400, 800\}$.

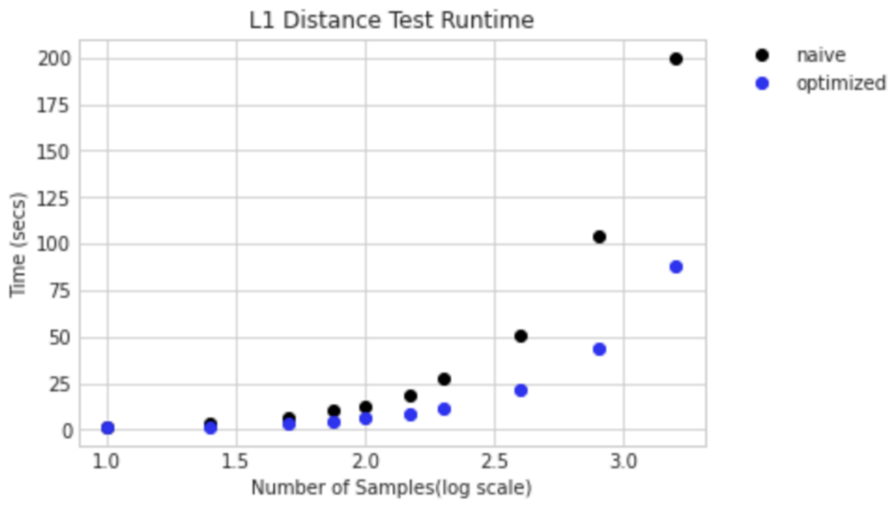


Figure 7: This is figure 6 with the number of samples in log scale.

Now, we evaluate the performance of algorithm 3 on our dataset. In both algorithms 2 and 3, we take m samples from distributions p and q multiple times. However, random sampling from a distribution can be quite expensive. In the naive approach, we generate samples within native Python loops. However, loops can be quite slow when processing such large datasets. This is due to the interpreter overhead of the loop itself.

Thus, in order to reduce sampling runtime, we use NumPy to vectorize sampling. We first replicate the discrete probability distribution m times. Then we generate random numbers with the same shape and magnitude as our distribution. And lastly, we shift distribution of each row according to the randomly generated numbers. The runtime boost of this approach is illustrated in figures 6 and 7. For $m \in [0, 1600]$, the optimized NumPy approach significantly outperforms the naive approach. When $m = 1600$, the runtime of naive sampling and optimized NumPy sampling are around 200s and 88s respectively.

```python
def native_loop(num_samples, sample_size, elements, probabilities):
    elements = list(elements) # because we later use .remove() method
    samples = []
    for _ in range(num_samples):
        sample = []
        candidate_elements = elements.copy()
        while len(sample) < sample_size:
            # choose an index as a candidate to include
            candidate_element = random.choice(candidate_elements)
            # decide whether to include in the sample or not
            if probs[candidate_element] >= random.random():
                sample.append(candidate_element)
                candidate_elements.remove(candidate_element)
        samples.append(sample)
    return samples
```

Listing 1: Naive Sampling

```python
import numpy as np
def multidimensional_shifting(num_samples, sample_size, elements, probabilities):
    # replicate probabilities as many times as ‘num_samples‘
    replicated_probabilities = np.tile(probabilities, (num_samples, 1))
    # get random shifting numbers & scale them correctly
    random_shifts = np.random.random(replicated_probabilities.shape)
    random_shifts /= random_shifts.sum(axis=1)[:, np.newaxis]
    # shift by numbers & find largest (by finding the smallest of the negative)
    shifted_probabilities = random_shifts - replicated_probabilities
    return np.argpartition(shifted_probabilities, sample_size, axis=1)[:, :sample_size]
```

Listing 2: Optimized Sampling using NumPy

However, one significant drawback of this approach is that it has a significantly higher memory complexity since we must replicate the discrete distribution m times. This drawback soon became evident as we increased m beyond 1600, where we started running into MemoryError. Because of the slow runtime of the naive implementation and the memory overhead in the NumPy implementation, we were not able to run meaningful experiments with larger samples since our domain sizes are on the order of hundreds of thousands.

However, one meaningful finding is that like in Section 2, determining the ℓ_1 distance between two distributions using brute force (figure 2) is actually faster than the sublinear ℓ_1-Distance-Test (algorithm 3). This is because in the brute force approach, our dataset is small enough such that we can store both distributions in hashmaps. On the other hand, in the sublinear ℓ_1-Distance-Test, we must sample from the two distributions many times. Since hashmap lookups are significantly faster than random sampling, it is expected that the brute force algorithm to have a better computational runtime performance.

One area of future work is to attempt other sampling algorithms to further optimize the sampling complexity. We did not have enough time to implement other sampling algorithms include weighted reservoir sampling [4] and fast inverse transform sampling [5].

4 Future Work

4.1 Unimodality Testing

We are also interested in testing the unimodality over the distribution of length of phrases over time. Some questions we can address through unimodality testing are:

- Do the length of memes tend to converge to a particular word count over time?
- What can we say about the attention span on online users over time?
- Are there relationships between meme lengths vs their popularity?

To begin, we define the collision count of S_I, the samples in an interval I to be $coll(S_I) = \sum_{i \in I} \binom{occ(i,S_I)}{2}$. We are interested in this definition because we can relate collision count to the ℓ_2 norm by the following lemma. We skip the proof of this lemma here. Refer to Section 4 of [2] if interested.

Lemma 9. *Let I be an interval and q be the conditional distribution of p on I, then*

$$(||q||^2 - \frac{\epsilon^2}{32|I|}) \leq \frac{coll(S_I)}{\binom{|S_I|}{2}} \leq (||q||^2) + \frac{\epsilon^2}{32|I|},$$

with probability at least $1 - O(log^{-3}(n))$ *if* $|S_I| = \Omega(\epsilon^{-4}\sqrt{|I|}loglog(n))$

The unimodality tester will use collisions in sample to determine a partition. A low count of collisions in the sample suggests a nearly uniform distribution of the weight. Thus, the statistic will result in a partition with close-to-uniform conditional distributions on each interval. After obtaining a partition, the tester will check if these close-to-uniform conditional distributions can be patched together into a unimodal flat distribution.

We describe the algorithm here:

Algorithm 4 : Unimodality-Test(p, ϵ)

```
1: $m \leftarrow O(\epsilon^{-4}\sqrt{n}log(n))$
2: Obtain $m$ samples $S$ from p. Start with the interval $I$ =[1,n], and bisect $I$ in half recursively as long as
       $$coll(S_I) \geq \frac{(1+\frac{\epsilon^2}{32})}{|I|}\binom{|S_I|}{2} \text{ and } |S_I| \geq \frac{m}{log^3(n)}$$
   Output FAIL if it performed more than $O(\frac{1}{\epsilon}log^2(n))$ splits.
3: Let $I_l = \{I_1, ..., I_l\}$ denote the partition of [1,n] into intervals induced by the leaves of the recursion from step 2
4: Obtain an additional sample $T$ of size $O(\epsilon^{-2}log^4(n))$
5: Let $hist(T, I_l)$ represent the unimodal flat distribution by $(w, I_l)$ where $w_j = \frac{occ(I_j,T)}{|T|}$
6: Output PASS if $hist(T, I_l)$ is $\frac{\epsilon}{2}$ close to a unimodal distribution. Output FAIL otherwise
```

Unimodality testing is quite similar to monotonicity testing covered in class. We skip the proof of algorithm 4 here. If interested, refer to Section 7 of [2].

Due to the size of the dataset, we could not run any meaningful experiments to test the unimodality over the distributions of length of phrases over time. This is because our new domain size is at most the maximum length of a common phrases, which is quite small for a normal sentence. However, we can using an alternative dataset such as the distribution of length of Wikipedia articles over time. This will increase our domain size to be at most the maximum length of a Wikipedia article. Using Wikipedia articles we can still learn about the attention span on online users over time and the relationships between article lengths vs their popularity? Hopefully, the domain size will be large enough such that our sublinear algorithms will be faster than brute force approaches.

4.2 Markov Chain

Random walks on Markov chains generate probability distributions over the states of the chain, induced by the endpoints of the random walks. We can apply ℓ_1-Distance-Test to test the mixing properties of Markov Chains. Refer to Section 4 of [1]. We can represent the distribution of phrases over a time period as a Markov state, and test the mixing properties of this Markov Chain.

5 Conclusion

In summary, to get a sense of the changes in the popularity of common phrases over time, we ran two sublinear algorithms on the distribution of phrases appearing in the Memetracker dataset. The first metric we looked at was ℓ_2 norm; we implemented an algorithm to compute this exactly, as well as an algorithm to compute this in sublinear time. We found in 2.1 that the distance in ℓ_2 norm squared tended to be on the order of 10^{-5}, and that it decreased as time progressed. There are many possible explanations for why this may occur. For example, it could be that specific phrases crop up and are used very frequently in correlation to high-profile events, such as the Olympic games or the presidential campaign season, and that language becomes more static in the absence of these types of events. Ultimately, we would need more information to conclusively determine why this occurred.

We then attempted to look at closeness in L1 norm, also in sublinear time. Unfortunately, this algorithm took too long to run and we were not able to infer any aspects of the distribution on phrases. Part of the reason is that in our implementation, random sampling is very expensive in terms of time complexity, instead of being constant as assumed in class. Despite the lack of results, we were able to see that vectorizing the sampling process can help with optimizing performance.

References

[1] Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *CoRR*, abs/1009.5397, 2010.

[2] Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 381–390. ACM, 2004.

[3] Siu-on Chan, Ilias Diakonikolas, Gregory Valiant, and Paul Valiant. Optimal algorithms for testing closeness of discrete distributions. *CoRR*, abs/1308.3946, 2013.

[4] Rajesh Jayaram, Gokarna Sharma, Srikanta Tirthapura, and David P. Woodruff. Weighted reservoir sampling from distributed streams. 2019.

[5] Sheehan Olver and Alex Townsend. Fast inverse transform sampling in one and two dimensions. 2013.