Instructor:
Prof. Nakul Verma
COMS W4771–Spring, 2022

Student:
Vishweshwar Tyagi
vt2353 @columbia.edu

Homework 1

Due: 11:59pm, March 07, 2022

Homework Problems

Solution-1:

(i) Upper sub-script will iterate over samples, lower sub-script will iterate over components, for example, x^i_k denotes the k^{th} component of the i^{th} sample. Without loss of generality, let each y^i be the disjunction of the first r components of x^i, i.e, $y^i = x^i_1 \vee x^i_2 \ldots \vee x^i_r \;\; \forall i \in [n]$ where $r \in [d]$ is fixed. Note that, $w(0) = \mathbf{1} \in \mathbb{R}^d$ is our initial weight vector.

Suppose $w(t)$ is our current weight vector, and our algorithm makes an error on a negative sample (x^i, y^i) with $y^i = 0$. Since $y^i = 0 \rightarrow x^i_k = 0 \;\; \forall k \in [r]$, and hence, the updated weight vector $w(t+1)$ will have $w_k(t+1) = w_k(t) \; \forall k \in [r]$. This means that, whenever our algorithm makes a mistake on a negative sample, the first r components of the updated weight vector are same as the the first r components of the current weight vector.

Now suppose $w(t)$ is our current weight vector, and our algorithm makes an error on a positive sample (x^i, y^i) with $y^i = 1$. Since $y^i = 1 \rightarrow \exists k \in [r]$ such that $x^i_k = 1$, and hence, $\exists k \in [r]$ such that $w_k(t+1) = 2w_k(t)$. Note that we must have $w_k(t) \leq d$ for mistake to happen, since otherwise $w(t) \cdot x^i \geq w_k(t) > d \rightarrow \hat{y}^i = y^i$, contradicting our assumption that model makes a mistake on (x^i, y^i). Also, $w(t) \cdot x^i \geq w_k(t)$ holds because w_j's are always positive due to the nature of updates and initialization.

This means that, whenever our algorithm makes a mistake on a positive sample, atleast one of the first r components of the current weight vector gets doubled but only those components among the first r components can get doubled that are $\leq d$.

To summarize, the first r components can only ever change when algorithm makes mistake on a positive sample, which forces atleast one of the first r components to get doubled. Moreover, any given component among the first r components can get doubled for atmost $1 + \log d$ times because $2^{\log d} = d \leq d$ (so this component could still contribute to one more mistake) but $2^{1+\log d} = 2d > d$ (won't contribute to any further mistakes). Therefore, our algorithm can make atmost $r(1 + \log d)$ mistakes on positive samples, because once these many mistakes have been made, we'd have each of the first r components equal to $2d$, and no further mistake would be made on a positive sample since $w \cdot x^j \geq 2d > d$ for all (x^j, y^j) with $y^j = 1$.

Therefore, $M_+ \leq r(1 + \log d)$

(ii) Let $TW(t) = \sum_k w_k$ and suppose algorithm makes mistake on a negative sample (x^i, y^i) with $y^i = 0$. This forces

$w_k \mapsto w_k/2 \ \ \forall k \in [d]$ such that $x_k^i = 1$, else $w_k \mapsto w_k$

Hence, the updated total weight becomes

$$\begin{aligned} TW(t+1) &= \sum_{k:\, x_k^i=1} \frac{w_k}{2} + \sum_{k:\, x_k^i=0} w_k \\ \rightarrow TW(t+1) - TW(t) &= \sum_{k:\, x_k^i=1} \big(\frac{w_k}{2} - w_k\big) \\ \rightarrow TW(t) - TW(t+1) &= \sum_{k:\, x_k^i=1} \frac{w_k}{2} > 0 \end{aligned} \tag{1}$$

Because RHS of (1) is > 0, weights decrease. Further, because our algorithm made a mistake on (x^i, y^i) with $y^i = 0$, this means that

$$\begin{aligned} w \cdot x^i &> d \\ \sum_k w_k x_k^i &> d \\ \sum_{k:\, x_k^i=1} w_k &> d \end{aligned} \tag{2}$$

From (1) and (2), we get,

$$\begin{aligned} TW(t) - TW(t+1) &= \sum_{k:\, x_k^i=1} \frac{w_k}{2} \\ \rightarrow TW(t) - TW(t+1) &> \frac{d}{2} \end{aligned} \tag{3}$$

as required.

(iii) Let $TW(t) = \sum_k w_k$ and suppose algorithm makes mistake on a positive sample (x^i, y^i) with $y^i = 1$. This forces

$w_k \mapsto 2w_k \ \ \forall k \in [d]$ such that $x_k^i = 1$, else $w_k \mapsto w_k$

Hence, the updated total weight becomes

$$\begin{aligned} TW(t+1) &= \sum_{k:\, x_k^i=1} 2w_k + \sum_{k:\, x_k^i=0} w_k \\ \rightarrow TW(t+1) - TW(t) &= \sum_{k:\, x_k^i=1} \big(2w_k - w_k\big) \\ \rightarrow TW(t+1) - TW(t) &= \sum_{k:\, x_k^i=1} w_k > 0 \end{aligned} \tag{4}$$

Because RHS of (1) is > 0, weights increase. Further, because our algorithm made a mistake on (x^i, y^i) with $y^i = 1$, this means that

$$\begin{aligned} w \cdot x^i &\leq d \\ \sum_k w_k x_k^i &\leq d \\ \sum_{k:\, x_k^i = 1} w_k &\leq d \end{aligned} \tag{5}$$

From (1) and (2), we get,

$$\begin{aligned} TW(t+1) - TW(t) &= \sum_{k:\, x_k^i = 1} 2w_k \\ \rightarrow TW(t+1) - TW(t) &\leq d \end{aligned} \tag{6}$$

Let $M_+,\ M_-$ be the total number of mistakes on positive and negative samples respectively. Suppose, $t = M_- + M_+$ From (3) and (6), we have,

$$TW(t) \leq \begin{cases} TW(t-1) - \frac{d}{2} & \text{if } t\text{'th mistake was on negative sample} \\ TW(t-1) + d & \text{if } t\text{'th mistake was on positive sample} \end{cases}$$

$$\rightarrow TW(t) \leq \begin{cases} TW(t-2) - 2 \cdot \frac{d}{2} & \text{if } t\text{'th and } (t-1)\text{'th mistakes were on negative sample} \\ TW(t-2) + 2d & \text{if } t\text{'th and } (t-1)\text{'th mistakes were on positive sample} \\ TW(t-2) - \frac{d}{2} + d & \text{if only one of } t \text{ or } (t-1)\text{'th mistake was on positive sample} \end{cases}$$

$$\cdots$$

$$\rightarrow TW(t) \leq TW(0) - \frac{d}{2} M_- + dM_+$$

Ofcourse, $0 < TW(t)$, since each component of the weight vector is always a power of 2. Using $TW(0) = d$ and (i), we get,

$$0 < TW(t) \leq TW(0) - \frac{d}{2} M_- + dM_+ \tag{7}$$

$$\rightarrow 0 \leq d - \frac{d}{2}(M_- + M_+) + \frac{3d}{2} M_+$$

$$\rightarrow 0 \leq 2 - (M_- + M_+) + 3M_+$$

$$\rightarrow (M_- + M_+) \leq 2 + 3r(1 + \log d) \tag{8}$$

as required.

Solution-2:

(a) We have,

$$\begin{aligned}
\phi(x) \cdot \phi(y) &= \sum_{i=0}^{\infty} \phi(x)_i \phi(y)_i \\
&= \sum_{i=0}^{\infty} \frac{2^i \gamma^i}{i!} \exp\big(-\gamma(x^2+y^2)\big)(xy)^i \\
&= \exp\big(-\gamma(x^2+y^2)\big) \cdot \sum_{i=0}^{\infty} \frac{1}{i!}\big(2\gamma \cdot xy\big)^i \\
&= \exp\big(-\gamma(x^2+y^2)\big) \exp(2\gamma \cdot xy) \\
&= \exp(-\gamma(x^2+y^2-2xy)) \\
&= \exp(-\gamma(x-y)^2) \\
&= \exp(-\gamma \cdot ||x-y||^2) \\
&= K(x,y)
\end{aligned}$$

as required.

(b) Let $g(x) : \mathbb{R}^d \to \mathbb{R}$ be a polynomial function.

We can write $g(x) = \sum_{i=1}^{m'} b_i x^{p_i}$ for some $m' > 0$, $b_1 \dots b_{m'} \in \mathbb{R}$ and $p_1 \dots p_m \in \mathbb{N} \cup \{0\}$.

Letting $m = \max\{p_i \mid i \in [m']\}$, we have $g(x) = \sum_{i=0}^{m} a_i x^i$ for some $a_1, \dots a_m \in \mathbb{R}$

Let $w = (w_0, w_1, \dots, w_t, w_{t+1}, \dots)$ be an infinite dimensional vector such that

$$w_j = \begin{cases} \sqrt{\frac{j!}{2^j \gamma^j}} \cdot a_j & \text{if } j \le m \\ \\ 0 & \text{if } j > m \end{cases}$$

We then have for $x \in \mathbb{R}$,

$$\begin{aligned}
\exp(\gamma x^2)\; w \cdot \phi(x) &= \exp(\gamma x^2) \cdot \sum_{j=0}^{m} \sqrt{\frac{j!}{2^j \gamma^j}} a_j \sqrt{\frac{2^j \gamma^j}{j!}} \exp(-\gamma x^2) x^j \\
&= \sum_{j=0}^{m} a^j x^j \\
&= g(x)
\end{aligned}$$

as required.

This way, since $\exp(\gamma x^2) \neq 0 \;\; \forall x \in \mathbb{R}$, we have $g(x) = 0 \iff \exp(\gamma x^2) w \cdot \phi(x) = 0 \iff w \cdot \phi(x) = 0$ which becomes linear since $w \cdot \phi(x) = 0$ is a linear in w.

(iii) Suppose we have a lot of training data and we know that best boundary is a high degree polynomial. If $\gamma > 1$ is large, γ^i will become very large for higher values of i, causing $w_k \to 0$ for higher values of k. Because of this, our model will not be able to learn the higher order coefficients of the high degree polynomial, and this reason along with presence of a lot of data will cause our model to underfit. So, in this case, we should keep $\gamma > 1$ relatively close to 1 in order to avoid underfit.

Now suppose we have very little data. If we keep γ large, this will ensure that our model weights $\to 0$ for higher order terms, which will ensure that our model doesn't learn complex higher order polynomials (these polynomials can potentially overfit our very little data). Hence, in this case, we can keep γ relatively large to avoid overfit.

Solution-3:

(i) The payoff matrix for p_1 is given by

$$\mathrm{P} = \begin{bmatrix} p_1/p_2 & | & \mathrm{r}_{p_2} & \mathrm{p}_{p_2} & \mathrm{s}_{p_2} \\ -- & -- & -- & -- & -- \\ \mathrm{r}_{p_1} & | & 0 & -1 & 1 \\ \mathrm{p}_{p_1} & | & 1 & 0 & -1 \\ \mathrm{s}_{p_1} & | & -1 & 1 & 0 \end{bmatrix}$$

(ii) The payoff matrix for p_2 is given by

$$\mathrm{Q} = \begin{bmatrix} p_1/p_2 & | & \mathrm{r}_{p_2} & \mathrm{p}_{p_2} & \mathrm{s}_{p_2} \\ -- & -- & -- & -- & -- \\ \mathrm{r}_{p_1} & | & 0 & 1 & -1 \\ \mathrm{p}_{p_1} & | & -1 & 0 & 1 \\ \mathrm{s}_{p_1} & | & 1 & -1 & 0 \end{bmatrix}$$

Clearly, $P = -Q$

(iii) Let $S \subset \{-1, 1\}^{n \times n}$ such that S has atleast one row containing only 1s, i.e, $\exists i \in [n]$ such that $A[i, :].T = 1_{n \times 1}$.

(iv) Let A be p_1's payoff matrix. We know that $A_{ij} \in \{\pm 1\}\ \forall i, j \in [2]$. Further, neither player has a move that results in a guaranteed win, which ensures that there is not a single row containing only 1s and there is not a single column containing only -1s. Since each entry can either be ± 1, we have that,

$$\mathrm{A} = \begin{bmatrix} p_1/p_2 & | & 1_{p_2} & 2_{p_2} \\ -- & -- & -- & -- \\ 1_{p_1} & | & 1 & -1 \\ 2_{p_1} & | & -1 & 1 \end{bmatrix} \text{ OR } \begin{bmatrix} p_1/p_2 & | & 1_{p_2} & 2_{p_2} \\ -- & -- & -- & -- \\ 1_{p_1} & | & -1 & 1 \\ 2_{p_1} & | & 1 & -1 \end{bmatrix}$$

Note that know any one entry in A fixes all other entries. Hence, A is uniquely determined by A_{11}. Also, p_1's gain is p_2's loss and vice versa, where their payoff sum to 0.

Clearly, the optimal strategy for p_2 is to choose the move which is different that the one chosen by p_1 if $A_{11} = 1$, or, choose the same move as p_1 if $A_{11} = -1$. More precisely,

Let $u_2(s^1, i)$ denote the expected payoff of p_2 when they decide to play move $i \in \{1, 2\}$ against p_1's strategy s^1, we have,

$$u_2(s^1, 1) = -A_{11}s_1^1 - A_{21}s_2^1$$
$$u_2(s^1, 2) = -A_{12}s_1^1 - A_{22}s_2^1$$

$$\rightarrow u_2(s^1, 1) - u_2(s^1, 2) = (A_{12} - A_{11})s_1^1 + (A_{22} - A_{21})s_2^1$$

$$\rightarrow u_2(s^1, 1) - u_2(s^1, 2) = \begin{cases} -2s_1^1 + 2_2^1 & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = 1 \\ -2s_1^1 + 2_2^1 & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = 1 \\ 2s_1^1 - 2_2^1 & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = -1 \\ 2s_1^1 - 2_2^1 & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = -1 \end{cases}$$

$$u_2(s^1, 1) - u_2(s^1, 2) \begin{cases} \leq 0 & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = 1 \\ > 0 & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = 1 \\ \geq 0 & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = -1 \\ < 0 & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = -1 \end{cases}$$

$$\rightarrow u_2(s^1, 1) \begin{cases} \leq u_2(s^1, 2) & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = 1 \\ > u_2(s^1, 2) & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = 1 \\ \geq u_2(s^1, 2) & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = -1 \\ < u_2(s^1, 2) & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = -1 \end{cases}$$

$$\rightarrow s^2 = \begin{cases} (0,\ 1)^T & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = 1 \\ (1,\ 0)^T & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = 1 \\ (1,\ 0)^T & \text{if } s_1^1 \geq 0.5 \text{ and } A_{11} = -1 \\ (0,\ 1)^T & \text{if } s_2^1 > 0.5 \text{ and } A_{11} = -1 \end{cases} \tag{1}$$

Clearly, s^2 defined above is deterministic depending on s^1 and A

(v) p_1's expected payoff will be

$u_1(s^1, s^2) = s_1^1 u_1(1, s^2) + s_2^1 u_1(2, s^2) = s_1^1\big(A_{11}s_1^2 + A_{12}s_2^2\big) + s_2^1\big(A_{21}s_1^2 + A_{22}s_2^2\big) = {s^1}^T A s^2$

Clearly, when p_1 plays with s^1 strategy, they will ensure themselves a payoff of atleast $\min_{s^2} {s^1}^T A s^2$. Hence, p_1's defensive strategy would be to look for s^1 that maximizes $\min_{s^2} {s^1}^T A s^2$. That is, an s^1 such that $\max_{s^1} \min_{s^2} {s^1}^T A s^2$.

Note that, $\min_{s^2} {s^1}^T A s^2 \leq \min_{j=1,2} u_1(s^1, j) \rightarrow \min_{s^2} {s^1}^T A s^2 \leq \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \quad \ldots (2)$

Further,

${s^1}^T A s^2 = \sum_{j=1}^{2} s_j^2 \big(\sum_{i=1}^{2} A_{ij} s_i^1 \big)$

$\rightarrow {s^1}^T A s^2 \geq \sum_{j=1}^{2} s_j^2 \big(\min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \big) = \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1$ because $\sum_{j=1}^{2} s_j^2 = 1$

Therefore, ${s^1}^T A s^2 \geq \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \rightarrow \min_{s^2} {s^1}^T A s^2 \geq \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \quad \ldots (3)$

From (2) and (3), $\min_{s^2} {s^1}^T A s^2 = \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \quad \ldots (4)$

Note that, a similar argument shows that, $\max_{s^1} {s^1}^T A s^2 = \max_{i=1,2} \sum_{j=1}^{2} A_{ij} s_j^2 \quad \ldots (6^*)$

$\big((6^*)$ will be used in part $(vii)\big)$

Therefore, the required optimization problem is:

$$\begin{aligned} & \max_{s^1} \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \\ \text{subject to} \quad & s_1^1 + s_2^1 = 1 \\ & s_1^1, s_2^1 \geq 0 \end{aligned} \tag{5}$$

which is equivalent to,

$$\begin{aligned} & \max_{s^1} z \\ \text{subject to} \quad & z \leq \min_{j=1,2} \sum_{i=1}^{2} A_{ij} s_i^1 \\ & s_1^1 + s_2^1 = 1 \\ & s_1^1, s_2^1 \geq 0 \end{aligned}$$

which is re-written as,

$$\begin{aligned} & \max_{s^1} z \\ \text{subject to} \quad & z \leq A_{11} s_1^1 + A_{21} s_2^1 \\ & z \leq A_{12} s_1^1 + A_{22} s_2^1 \\ & s_1^1 + s_2^1 = 1 \\ & s_1^1, s_2^1 \geq 0 \end{aligned} \tag{6}$$

(vi) In (6), we can instead minimize $-z$, which gives the required formulation:

$$\begin{aligned} & \min_{s^1} -z \\ \text{subject to} \quad & z \leq A_{11}s_1^1 + A_{21}s_2^1 \\ & z \leq A_{12}s_1^1 + A_{22}s_2^1 \\ & s_1^1 + s_2^1 = 1 \\ & s_1^1, s_2^1 \geq 0 \end{aligned} \tag{7}$$

$z \leq A_{1j}s_1^1 + A_{2j}s_2^1$ are convex sets in $\mathbb{R}^2$ because $A_{1j}s_1^1 + A_{2j}s_2^1$ is linear in $s^1 \ \ \forall j \in \{1, 2\}$

$s_1^1 + s_2^1 = 1$ is again convex set, because these points are on a straight line.

$s_1^1, s_2^1 \geq 0$ together define the non-negative quadrant, which is convex.

The objective function $f(s^1) = -z$ is independent of s^1, and hence, is trivially convex.

Thus, the feasible region is an intersection of convex sets, therefore is convex too. We showed that the objective is also convex. Therefore, (6) is a convex optimization problem in s^1.

(vii) From previous question, we have that (7) is equivalent to

$\max_{s^1} \min_{s^2} {s^1}^T A s^2 = \min_{s^2} \max_{s^1} {s^1}^T A s^2 = \min_{s^2} \max_{s^1} {s^1}^T A s^2 = \min_{s^2} \max_{i=1,2} \sum_{j=1}^{2} A_{ij}s_j^2$ where the last equality follows from (6*) and we interchange max min to take the dual.

The last problem can be reformulated as:

$$\begin{aligned} & \min_{s^2} z \\ \text{subject to} \quad & z \geq \max_{i=1,2} \sum_{j=1}^{2} A_{ij}s_j^2 \\ & s_1^2 + s_2^2 = 1 \\ & s_1^2, s_2^2 \geq 0 \end{aligned}$$

which is re-written as,

$$\begin{aligned} & \min_{s^2} z \\ \text{subject to} \quad & z \geq A_{11}s_1^2 + A_{12}s_2^2 \\ & z \geq A_{21}s_1^2 + A_{22}s_2^2 \\ & s_1^2 + s_2^2 = 1 \\ & s_1^2, s_2^2 \geq 0 \end{aligned}$$

Solution-4:

(i) Sequential context associated with a sentence is not captured with unigram vectorization. For example - "The weather was good but the restaurant was horrible" would have the same representation as "The weather was horrible but the restaurant was good", but clearly the first sentence has a negative sentiment towards the restaurant (and hence should be labeled 0 by our classifier) whereas the second sentence has positive sentiment towards restaurant (and should be labeled 1). But our classifier can only predict both of these sentences as either both 0 or both 1 since the representation in which we're going to feed both these sentences to our classifier is exactly the same, which is a consequence of unigram vectorization.

Other issues include extremely long vectors that are sparse. Furthermore, these vectors also fail to capture linguistic similarity between words, or detect sarcasm.

(ii) TF unigram train scores: accuracy:0.889, precision:0.906, recall:0.929

TF unigram test scores: accuracy:0.888, precision:0.905, recall:0.928

TF-IDF unigram train scores: accuracy:0.890, precision:0.905, recall:0.932

TF-IDF unigram test scores: accuracy:0.889, precision:0.904, recall:0.931

TF unigram bigram train scores: accuracy:0.904, precision:0.919, recall:0.938

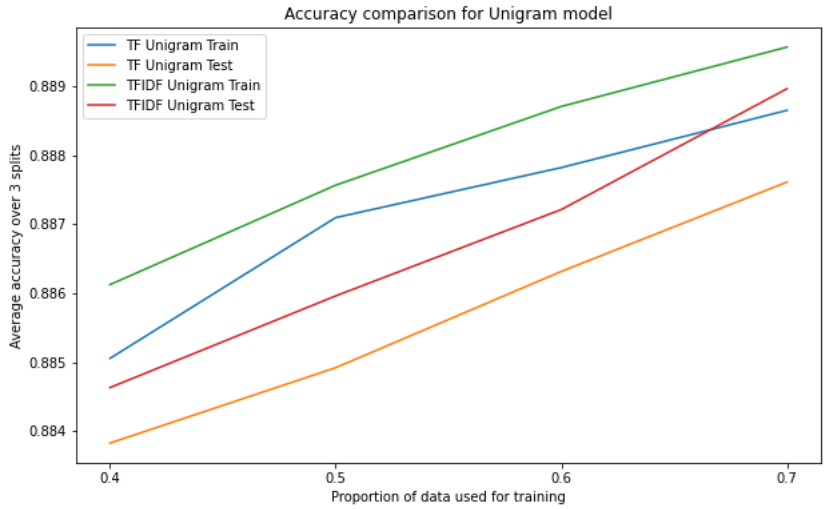
TF unigram bigram test scores: accuracy:0.902, precision:0.917, recall:0.936

TF-IDF unigram bigram train scores: accuracy:0.905, precision:0.918, recall:0.941

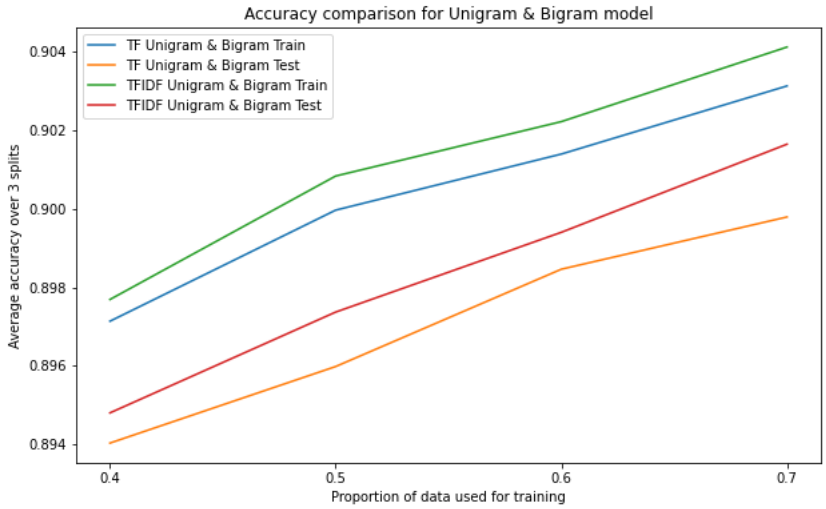
TF-IDF unigram bigram test scores: accuracy:0.902, precision:0.916, recall:0.939

Given the size of training and test data, we can conclude a significant improvement over using unigrams and bigrams as compared to only using unigrams. Further, we also notice a slight improvement when using term-frequency inverse-document-frequency vectorization as compared to only term-frequency vectorization.

(iii) Following are the observations as a result of varying training data size and for each training size, we averaged over 3 splits.

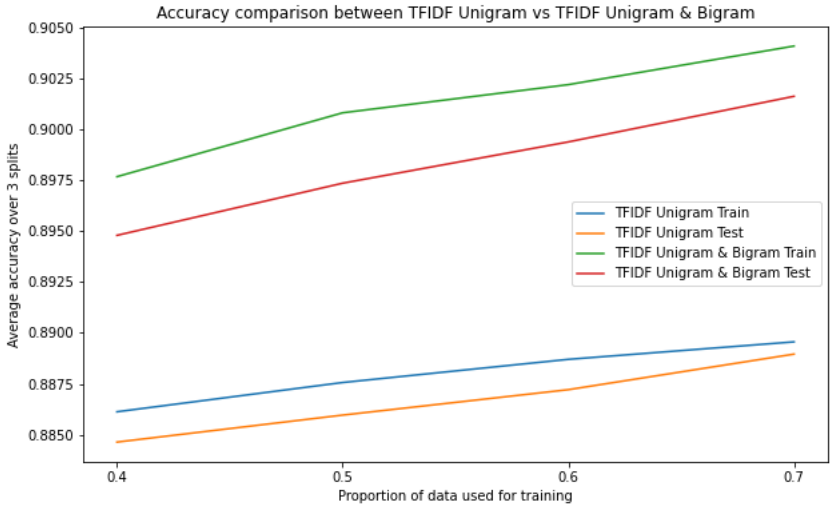


We see that among the unigram models, TFIDF performed the best.



We again see that among the unigram and bigram models, TFIDF performed the best.

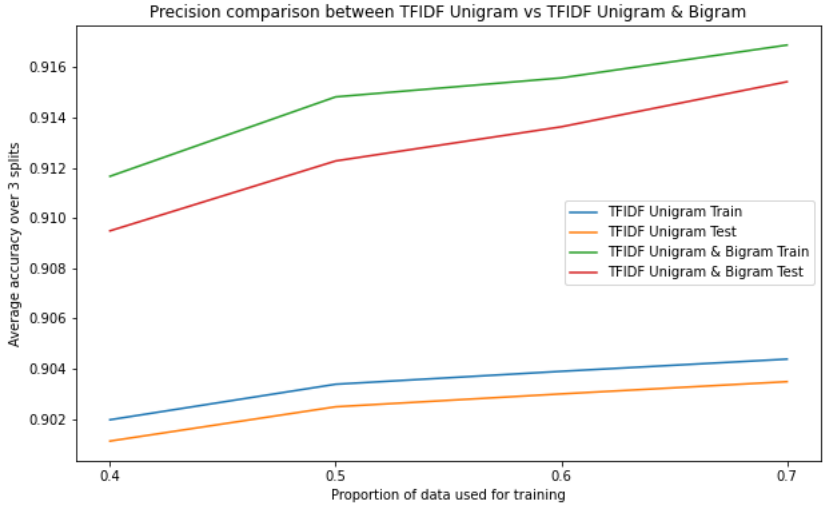
Now, let us compare TFIDF Unigram Vs TFIDF Unigram and Bigram

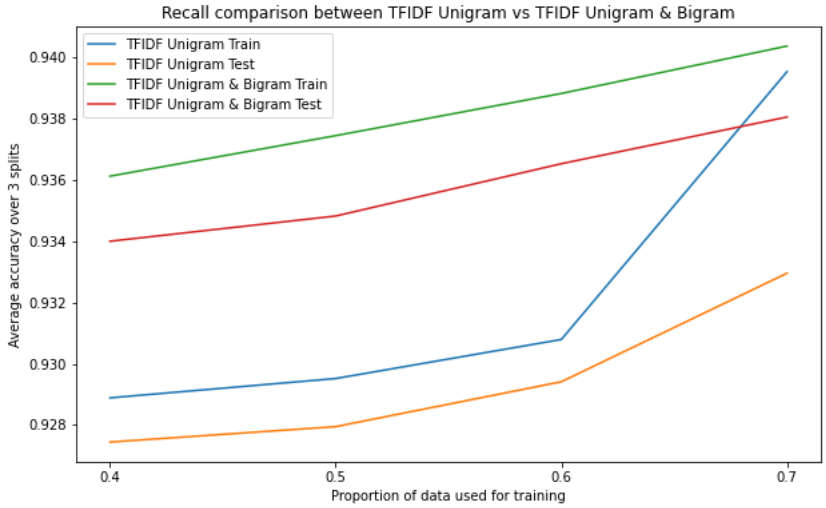


We see that TFIDF Unigram and Bigram performs significantly better than its counterpart. Our best model in terms of accuracy, therefore, is TFIDF Unigram and Bigram.

Thus, incorporating bigrams along with unigrams definitely helps, possibly in adding more context to the vectorized representations.

Let us also compare their precision and recall





Clearly, TFIDF Unigram and Bigram wins with respect to all metrics: accuracy, precision and recall.

Conclusion: We conclude that Term-frequency Inverse-document-frequency with unigrams and bigrams is our best model.

(iv) TF unigram model's 10 words corresponding to most negative weights (in order of increasing weights):

['worst', 'lacked', 'flavorless', 'mediocre', 'tasteless', 'disgusting', 'meh', 'hopes', 'disappointing', 'ruined']

TF unigram model's 10 words corresponding to most positive weights (in order of decreasing weights):

['perfection', 'heaven', 'gem', 'disappoint', 'heavenly', 'phenomenal', 'incredible', 'perfect', 'superb', 'perfectly']

TF-IDF unigram model's 10 words corresponding to most negative weights (in order of increasing weights):

['worst', 'mediocre', 'bland', 'flavorless', 'awful', 'horrible', 'tasteless', 'disappointing', 'meh', 'lacked']

TF-IDF unigram model's 10 words corresponding to most positive weights (in order of decreasing weights):

['delicious', 'perfection', 'amazing', 'perfect', 'great', 'excellent', 'awesome', 'heavenly', 'gem', 'fantastic']