1 Basic n-gram Models

1.1 Generating Shakespeare

We train some unsmoothed, uninterpolated n-gram models on some Shakespeare corpus, including *The Tragedy of Coriolanus*, *King John*, and *The Tempest*. Our results are below:

n	**Output**
2	Fir ling I hichantrablumbe dile die the you, suess now, I thater, itheaccow: Biseer my leet its, likere me heme, BIROSTOLUS A sh'd Save sine in ase ot I sell dur'd he paptres, sil terwillseeck thall meou chu Tomere? I a verer, butur mad tood, for f
3	First, Breat duke do your flat; lesoldinna, your lead yountenbertain? AJAX: Now ther welver my thosed, the in that's the ear, I be thee Duke speak nown rageonation therer: Fare speach you wondire, whold by, Rebell In sore nightertainst eased with of
4	First Lord of Amiens him. Henry in this know, my riotous, let mannerly will choked his out of all cut. For that your fight thou are captainly grandam's sleep out on. Now, Staff heavy at had his once; And so, I do requirrevolt To yield to march shall
7	First Clown: So study of this fellow the youthful Troilus! the prince him of his books, or with spots you shalt never said to speak; if savages,–as some know not what. SHALLOW: It appear, too suddenly enrapt To give way to herself, The wish'd to hi

Table 1: Generated Shakespeare Text

From the above, we can see that as n increases, the passages become more coherent. (When $n = 2$, the text generated is almost reminiscent of German due its lack of coherence.) This is because the model is based on generating text character-by-character, rather than word-by-word, so for small n the words are unlikely to make any sense. Furthermore, the text generated by $n = 7$ is the most similar to Shakespeare, and we can see elements of Shakespeare present in the text, such as "Troilus" and "SHALLOW:," which is a character in *Henry IV, Part 2* and *The Merry Wives of Windsor*. In fact, it is likely that at this high value for n, our model is just copying Shakespeare verbatim. Thus, these n-gram models heuristically appear better than 1000 monkeys working at 1000 typewriters.

We also note that all of the passages we generated start with "F." We hypothesize that this is due to the fact that the first word in the training file is "First," so it is the only word padded with tildes. Thus, since

the first context given is always the pad, the first word generated is always "First" when the value of n is large enough.

1.2 Generating Shrek

For fun, we tried generating text using the script from Shrek (2001).[1]

n	**Output**
2	{Muffeep -Take yon {Gas befor hadell you ors. Shrove, are orgesion, "Grund my my lou'res quesn't unces. I'm a lover thad -But alk. I hat band pre bodere you lind Father's ha! Anyoustell don' oh-hut -You low whey! Aaway, em." wo! Baby, don't I
3	{Man, shoo! {Gasps, by about the haven alway. -Hey. -Yeah, yeah, face I means hole ture numb I with the stairy noble for I doing here. -Hey, build any out. Squess Fiona? -Take it? -So, yes a broans have keep back! Thumbling an thout no,

[1] A transcript of the movie was found at script-o-rama.com/movie_scripts/s/shrek-script-transcript-mike-myers.html

4	{Man's kiss. -I'm look in a quest. -You know! This long? Like the part is ever see the shilliona! I'll gets small. Believe People betters} {Creak a slight. Yeah, would we kind us late Just look so if I tell on think? I want to her last know, y
7	{Man} Once upon a time. -Hey, hey, come one. All you have to do was ask. Okay? -Well, yes. {Laughing} -What? -That's- - Is that sunset. -Sunset? -Oh, no, No! Screams Screams Whistles -I guess we better idea. You and thought of you

Table 2: Generated Shrek Text

1.3 Generating Bee Movie

We also tried generating text using the scrip from the *Bee Movie* (2007)[2].

[2]www.script-o-rama.com/movie_scripts/a1/bee-movie-script-transcript-seinfeld.html

n	**Output**
2	Actallen But. I'm ta lor laded can. Whe you quishoney, makfacke? Hexion How. Amende. You knarry. Lio. Mare gooke cantong thad that a le some he ase ou a parry wit one? - Yousesn't unnybeithe ging an. How dive cout ussugh as thing thou kid wit! The a
3	Accorry leave you do your fligh you detale! You do. - You want to flow, I knew thing. Yes? Barry. Wait, the have Turname! Easy, be are shalf. That chese race? Suppossionstake poing one you up to sory present that take time. Do would you discussive! H
4	According! I'm gonna guest enough of the right. You're from the just a tonio with pollen the mattery. - Six miles it! I courself in a lot out! Pound, already for the more bug. He have the ballow, by a walls? Oh, my. That wouldn't? I'd know we gonna t
7	According to me! You had your Uncle Oarl was on that plane. All I needed was a gift. Once at the flower. - OK. You get a nurse to close the court and stall. Stall any way you can'tjust decide? Bye. I just can't get these. Milk, cream, cheese, it's al

Table 3: Generated Shrek Text

We note that for both the Bee Movie and Shrek, as n increases, eventually we have that the first word of the generated text is the same as the first word of the input text (for the same reasons described above).

2 Perplexity, Smoothing, and Interpolation

Next, we try improving our initial N-gram model by adding perplexity, smoothing, and interpolation. We compared the perplexity of a New York Times Article, a selection of Shakespeare's sonnets, and the Shrek and Bee Movie scripts to the original training dataset of Shakespeare's plays used previously. To calculate these perplexities, we concatenate all paragraphs first. We first fix $n = 2$ and vary k from 1 to 4. For the interpolated models, we weigh λ equally (so $\lambda_i = \frac{1}{n+1}$, where $i \in [0, n]$).

Text Type	n	k	**Uninterpolated**	**Interpolated**
New York Times Article	2	1	11.019	13.026
New York Times Article	2	2	10.981	13.031
New York Times Article	2	3	10.991	13.050
New York Times Article	2	4	11.018	13.072
Shakespeare Sonnets	2	1	7.910	10.329
Shakespeare Sonnets	2	2	7.932	10.370
Shakespeare Sonnets	2	3	7.966	10.409
Shakespeare Sonnets	2	4	8.004	10.446
Shrek	2	1	13.477	14.541
Shrek	2	2	13.218	14.524
Shrek	2	3	13.110	14.538
Shrek	2	4	13.057	14.560
Bee Movie	2	1	10.102	12.756
Bee Movie	2	2	10.116	12.819

Bee Movie	2	3	10.158	12.875
Bee Movie	2	4	10.208	12.925

Table 4: Perplexities of Various Texts to Shakespeare's Plays (Fixed n)

As we can see above, the perplexity for Shakespeare Sonnets is much lower than the perplexity for the other three texts, indicating that the Shakespeare Sonnets are similar to Shakespeare's plays, which is to be expected since the Shakespeare sonnets and plays are by the same author and time period. Furthermore, as k increases, the perplexity decreases. The perplexity for the interpolated models is also higher than the perplexity for the uninterpolated models.

Next, we fix $k = 1$ and vary n. For the interpolated, the λ's are still equal. Our results are below:

Text Type	n	k	**Uninterpolated**	**Interpolated**
New York Times Article	2	1	11.019	13.026
New York Times Article	3	1	9.789	10.950
New York Times Article	4	1	10.852	10.150
New York Times Article	7	1	34.088	12.756
Shakespeare Sonnets	2	1	7.910	10.329
Shakespeare Sonnets	3	1	6.122	8.189
Shakespeare Sonnets	4	1	6.486	7.379
Shakespeare Sonnets	7	1	21.690	10.588
Shrek	2	1	13.477	14.541
Shrek	3	1	11.556	12.080
Shrek	4	1	13.150	11.254
Shrek	7	1	34.319	9.917
Bee Movie	2	1	10.102	12.756
Bee Movie	3	1	8.977	10.588
Bee Movie	4	1	10.870	12.248
Bee Movie	7	1	32.808	11.104

Table 5: Perplexities of Various Texts to Shakespeare's Plays (Fixed k)

With a fixed k, we can see that once again, the perplexity for Shakespeare Sonnets are much lower than the other texts, further supporting the fact that the Shakespeare sonnets are most similar to the Shakespeare plays of the four texts. The interpolated perplexities are also higher than the uninterpolated perplexities on average, with the exception of when $n = 7$. Interestingly, the uninterpolated perplexity spikes when $n = 7$.

Next, we experiment by varying λ. We tried an even weighting, a heavier λ on smaller n-gram models, and heavier weighting on larger n-gram models. We suspect that the perplexity of the interpolated model with larger λ for larger n-gram models to be smaller, since more of the context is considered.

Text Type	λ	**Perplexity**

New York Times Article	[0.25, 0.25, 0.25, 0.25]	10.950
New York Times Article	[0.4, 0.3, 0.2, 0.1]	12.825
New York Times Article	[0.1, 0.2, 0.3, 0.4]	9.826
Shakespeare Sonnets	[0.25, 0.25, 0.25, 0.25]	8.189
Shakespeare Sonnets	[0.4, 0.3, 0.2, 0.1]	10.044
Shakespeare Sonnets	[0.1, 0.2, 0.3, 0.4]	7.072
Shrek	[0.25, 0.25, 0.25, 0.25]	12.080
Shrek	[0.4, 0.3, 0.2, 0.1]	14.194
Shrek	[0.1, 0.2, 0.3, 0.4]	10.937
Bee Movie	[0.25, 0.25, 0.25, 0.25]	10.588
Bee Movie	[0.4, 0.3, 0.2, 0.1]	12.617
Bee Movie	[0.1, 0.2, 0.3, 0.4]	9.381

Table 6: Perplexities of Various Texts to Shakespeare's Plays ($n = 3, k = 1$)

The results above support our initial hypothesis.

We also note that across all three tables, we note that we always have that the perplexities of the texts in increasing order for the same movies is the following: Shakespeare Sonnets, Bee Movie, New York Times Article, Shrek. This suggests that varying n, k, and λ's change the resulting perplexity, but are monotonic transformations.

3 Text Classification

For the text classification task, we tried both the uninterpolated N-gram model and the interpolated N-Gram model, both with varying values for n and k (and distribution of λ weights for the interpolated model). Our results are below. In general, our strategy for discovering the best-performing interpolated model was first finding the accuracy of the uninterpolated models on the validation set and using this information to tweak the parameters.

3.1 Uninterpolated Model Discussion

The best models for among our uninterpolated models were with the following values of parameters n and k: $(1, 0), (1, 1), (1, 2), (2, 1), (2, 2), (3, 1)$, and $(3, 2)$. While models with the following values for n and k performed the worst: $(0, 0), (0, 1), (0, 2), (3, 0), (4, 0), (5, 0), (5, 2)$.

Our results show that higher values of n along with a zero-value of n are harmful to the performance of the uninterpolated model. We suspect that this is because a zero-value for n lacks information regarding the different forms of city-names in different countries, and higher values of n are over-fitted to the training set.

Additionally, we find that without smoothing, our models severely underperform. This is to be expected since without add-k smoothing, our models likely struggle under novel characters/contexts.

We submitted the test label predictions for one of our best uninterpolated models ($n = 3$ and $k = 1$) and got an accuracy of roughly 0.67.

Model	n	k	**Validation Accuracy**
Uninterpolated	0	0	0.49
Uninterpolated	0	1	0.49
Uninterpolated	0	2	0.49
Uninterpolated	1	0	0.64
Uninterpolated	1	1	0.65
Uninterpolated	1	2	0.64
Uninterpolated	2	0	0.51
Uninterpolated	2	1	0.66
Uninterpolated	2	2	0.66
Uninterpolated	3	0	0.27
Uninterpolated	3	1	0.66
Uninterpolated	3	2	0.65
Uninterpolated	4	0	0.18
Uninterpolated	4	1	0.60
Uninterpolated	4	2	0.57
Uninterpolated	5	0	0.14
Uninterpolated	5	1	0.53
Uninterpolated	5	2	0.51

Table 7: Results for Uninterpolated Model

Model	n	k	λ's	**Validation Accuracy**
Interpolated	2	1	(0, 0.5, 0.5)	0.68
Interpolated	2	2	(0, 0.5, 0.5)	0.67
Interpolated	3	1	(0, 0, 0.5, 0.5)	0.69
Interpolated	3	1	(0, 0, 0.6, 0.4)	0.69
Interpolated	3	1	(0, 0.4, 0.3, 0.3)	0.69
Interpolated	3	2	(0, 0, 0.5, 0.5)	0.68
Interpolated	3	2	(0, 0, 0.6, 0.4)	0.68
Interpolated	3	2	(0, 0.3, 0.4, 0.3)	0.69

Table 8: Results for Interpolated Model

3.2 Interpolated Model Discussion

For experimenting with our interpolated models, as mentioned before, we referred to our results for the uninterpolated models to decide tweak our parameters. This led us to trying models with the parameters listed in Table 8.

Seeing as our uninterpolated models with parameters $n = 1, 2, 3$ seemed to generally perform the best with $k = 1, 2$. We trained several different interpolated models with values of lambda reflecting the relative performance of these uninterpolated models. In all of our models we let $\lambda_0 = 0$ since the uninterpolated models with $n = 0$ generally performed very poorly, so we decided to give it no weight in the interpolated models.

In the end, many of our interpolated models performed very similarly—most notably, our models with (n, k, λ) of $(3, 1, (0, 0, 0.5, 0.5)), (3, 1, (0, 0, 0.6, 0.4)), (3, 1, (0, 0.4, 0.3, 0.3))$, and $(3, 2, (0, 0.3, 0.4, 0.3))$. Submitting the test label predictions for these models only improved our accuracy of our uninterpolated model slightly with an accuracy of roughly 0.69 for the best performing of these interpolated models with parameters $(3, 2, (0, 0.3, 0.4, 0.3))$.

Below, we compiled a sample of some the misclassified cities from this best performing model under the validation set.

City Name	**True Label**	**Pred Label**
sefidbid-e ‘olya	af	ir
kirn	de	ir
deqing xian	de	cn
zinst	de	in
myntbole	fi	za
fall	fi	fr
kerbouzard	fr	za
jandwa	in	pk
sitio porteira da tradicao	ir	de
santa rosa de siria	ir	de
lapvandan	ir	fr
ragiti	ir	in
tamruta	pk	cn
eselfontein	za	fr
qhoehane	za	ir
fienile l’ avanzarola	za	fi

Table 9: Misclassified Countries Under Best Interpolated Model

There does not seem to be a discernable pattern to the flaws of our model. We do note, however, that a large majority of the mislabeled cities have true labels of de (German), ir (Iran), and za (South Africa). In general, we would expect our model to mislabel cities with countries that have similar languages as the true label, but our sample of errors shows this to not be the case. We also noticed that the true label of “deqing qian” was de (German), which is clearly mislabeled, and our model actually correctly predicted the cities true home-coutry, China (cn). While this classification task does not seem to difficult at first glance, examining the cities our model mislabeled more closely, we find that this task is much more difficult than we initially believed it to be, and are therefore satisfied with the overall performance of our best model.