Candidate Number: 237707
Word Count: xxx

Advanced Natural Language Processing Coursework

Introduction

In recent years, major breakthroughs in the internet of thing (IOT) and computational linguistics have provided researchers with a growing number of tools and resources to access text data as well as various means to effectively deploy language models to tackle numerous challenges (Verma et al., 2015) such as next word prediction. With an ever-increasing size of vocabulary and the evolution of language over time, concepts can often be conveyed in many different ways (Huang et al., 2013). One of the biggest challenges in NLP is achieving the concrete understanding of the evolving changes in semantics (Mitra et al., 2015). Therefore, lexical matching necessitates the endless examination and interpretation of text. This limitation signifies the relevance of the text data selected to successfully carry out semantic modeling.

The Microsoft Sentence Completion Challenge (Zweig and Burges, 2011) was created with an aim to provide an adequate framework to compare various semantic modelling techniques. Participants are provided with a training dataset of over 500 novels from the 19^{th} century (acquired from the Project Gutenberg collection). The task is to devise a language model that is able to predict the missing words in a test set of 1040 sentences extracted from five of Sir Arthur Conan Doyle's novels. For each sentence in the test dataset, contestants are provided with 5 choices of words to fill in the blank (comparable SAT-style questions). The correct choices are represented by the corresponding words that actually appear in the novel. On the other hand, the four other choices are devised using "imposter sentences" intending to increase the difficulty of this task by extracting words with similar occurrence statistics.

The MRSCC (Zweig and Burges, 2011) was designed to provide imposter words that may at times produce adequate completion of sentences. Various procedures were put in place to ensure the correct choices are clearly identifiable as the most likely ones. A maximum entropy n-gram model was deployed with the threshold set at 10^{-4} in order to select infrequent words. The two words on the left context of the target words were utilized to extract 150 alternates that adhered with the frequency threshold. Thereafter, sentences that did not contain the target word were scored and the top 30 were retained. Finally, a degree of "human grooming" was also delegated to ensure grammatical correctness, reduce ambiguity and remove profanity.

This paper will take on the MRSCC. Firstly, a unigram and bigram will be used as the baseline models. The base line model will then be scaled with construction of trigram and quadgram. Finally, a word similarity model (Word2Vec) will be implemented and compared with the baseline models as well as the n-gram models.

2. Method

2.1- Related Work

Zweig and Burges (2011) implemented three baseline models. The first model combined three simple n-gram models (bigram, trigram and quadgram). This combined approach saw each n-gram model making a prediction for each choice provided in the testing data using weighed scores that corresponds to the number of context windows used (bigram=1, trigram=2 and quadgram=4). Similarly, the second model endorsed the same approach but also included word frequency thresholding as well as smoothing. Word frequency thresholding was used to remove ambiguous vocabulary that appeared less than four times. On the other hand, the implementation of Good Turing smoothing effectively estimated the probabilities of unknown words using probabilities of words that only appeared once. The third and final model was based on latent semantics analysis where each sentence was perceived as a document and words were represented in 300-dimensional vectors. Thereafter, cosine similarity measure was performed and the choice with the highest average cosine similarity is selected as the final prediction.

2.2- N-gram Models

An n-gram is a sequential model that assigns a probability for each target word based on their respective frequencies in a given a context window. This method effectively provides a set of co-occurring words and their probabilities. The total number of words captured in the context and target windows accounts for the n value in the n-gram model [fig. 1]. While n-gram models can be advantageous in in estimating likelihood by encoding word sequences, they are limited to only focus on local context (Woods, 2016) and may be disadvantageous in capturing the full scope of sentences.

Sample Sentence	**"The cat sat on the mat"**	
1/Uni gram (baseline)	Context = "The" → Probability	
2/Bi gram (baseline)	Context = "The"	Target(s): "cat" → Probability
3/Tri gram	Context = "The cat"	Target(s): "sat" → Probability
4/Quad gram	Context = "The cat sat"	Target(s): "on" → Probability

2.2.1- Pre-processing and Tokenization

As a starting point, each word in the sentences retrieved from the training data will be tokenized and embedded with a "__START" and "__END" at each end. Subsequently, Case normalization will be used to ensure that the context windows do not include duplicates. Similarly, numbers will also be replaced to "__NUM" effectively aiming to reduce the number of ambiguous tokens recorded in the context window. For example, "I work out 2 times per week" and "I work out 10 times per week" would increase the probability of target word "times" while maintaining contextual information.

2.2.2- Smoothing and Frequency Thresholding

In some cases, n-gram models come across words that have not appeared frequently enough to yield concrete results. These words are represented by the percentage of times they appeared in the corpus also known as the Out of Vocabulary (OOV) rate (Jurafsky and Martin, 2021). Using frequency thresholding, OOV word are designated as "__UNK" and their total frequencies are stored accordingly.

Smoothing deals with unseen words that appear in the training set. Church and Gale (1991) demonstrated that apart from words frequencies of less that 2 in a bigram model, estimation would remain accurate following the discounting/subtraction of 0.75 from their respective frequencies. According to Jurafsky and Martin (2021), the interpolated Kneser-Ney (KN) algorithm is regarded as the best performing smoothing technique used on n-gram models. Rooted in the absolute discounting method devised by Church and Gale (1991) which subtracts the discount values from n-gram frequencies, KN smoothing is an augmented approach that better handles lower order unigram distribution (Jurafsky and Martin, 2021). KN smoothing technique can be represented by the following equation:

$$\boldsymbol{pKN}\left(\boldsymbol{w_i}\,\middle|\,\boldsymbol{w_{i-n+1}^{i-1}}\right) = \frac{\max\{c(w_{i-n+1}^{i}) - \delta, 0\}}{\sum_{wi}\, c(w_{i-n+1}^{i-1})} + \frac{\delta}{c(w_{i-n+1}^{i})}\boldsymbol{N_i} + (\boldsymbol{w_{i-n+1}^{i-1}}\,\bullet)\,\boldsymbol{pKN}\left(\boldsymbol{w_i}\,\middle|\,\boldsymbol{w_{i-n+2}^{i-1}}\right)$$

2.2.3- Implementation and Hyperparameter Tuning

Tokenization → Tokenize the training data using a list of lists that include pre-processed tokenized words from the sentences in the corpus.
Frequencies → Storing frequencies for each n-gram model accordingly in a dictionary.
Thresholding → Designate uncommon words and store the sum of their frequencies.
KN → Store the smoothed values using a discount rate of 0.75
Probability → Convert the frequencies to probabilities for n-gram models as well KN.
Prediction → The probabilities of the word choices in the testing data given the context window are calculated starting from the left context and move one element at a time (where possible). These probabilities are then multiplied (where the number of probabilities computed is proportionally larger

given the increasing of the context window/n-gram). The words with maximum probability from choices is then selected as the final prediction.

Following implementation, various experimentation will aim to find the optimal parameters to maximize the performance of the model. As illustrated by Church and Gale (1991) a value of 0.75 is suitable for words that appear at least 2-9 times in a bigram model. However, ranges from 2 to 5 will be implemented as a frequency threshold in consideration of trigram and quadgram which may perform poorly given a high threshold. Additionally, different samples sizes of the training data to observe the behavior of the n-gram models in response to the hyperparameters and identify the optimal parameters for each model.

2.3- Similarity Model (Word2Vec)

In contrast the n-gram models which focuses on the local context, similarity models present a different method which can capture wider contexts of sentences as oppose to just focusing on co-occuring words. Word embedding is a numeric input vector that represents a word. In short, it can approximate meaning with the distributional assumption that similar words are used together more often. For instance, "nurse" and "doctor" will be used together more often with "hospital." While capturing the contextual similarities between words, it offers a more efficient and articulate approach that is composed of a dense vector representation. However, besides being highly corpus dependent, limitations include shortcomings in respect to their capacities indistinguishing homophones.

Word2Vec is a highly regarded algorithm that produces word embedding and is known to outperform traditional methods such as Pointwise Mutual Information (Naili et al., 2017). It is enabled by one/two neural models and consists of two distinct architectures known as Skip Gram (SG) and Continuous Bag of Words (CBOW).

2.3.1- Preprocessing

Mikolov et al. (2013) suggested that frequently occurring tokens (i.e. stop words or punctuations) provide less contextual information. This resonates with the idea that word embedding models are less reliant on sequential understanding of the context. Therefore, digit, punctuation as well as stop word removal may be applied in addition to case normalization as a preprocessing step.

2.3.2- Skip Gram vs Continuous Bag of Words

CBOW is able to predict the current word based on a given context. Firstly, the context goes into the input layer. It then proceeds with the transformation of each word as s vector representation. This is referred to as the hidden layer. While the CBOW algorithm is known to perform better with frequent words, the SG algorithm is known to better handle smaller training samples with infrequent word. Furthermore, the SG algorithm's input is imperative of the target word while the output represents the context. Meanwhile, both models draw comparisons upon vectorized word inputs and outputs effectively exploiting backpropagation or stochastic gradient decent to maximize the following objective functions (Naili et al., 2017):

CBOW	SG
$\frac{1}{V}\sum_{t=1}^{v} \log\left(p\left(\frac{m_t}{m_{t-\frac{c}{2}} \dots m_{t+\frac{c}{2}}}\right)\right)$	$\frac{1}{V}\sum_{t=1}^{v}\sum_{t=t-c, j\neq t}^{t+c} \log\left(p(m_j \mid m_t)\right)$

Although both algorithms face limitations in respect to time complexity in the training process, Mikolov et al. (2013) proposed Hierarchical SoftMax (HS) and Negative Sampling to circumvent this problem. Based on the Huffman Tree, the HS algorithm is based on a binary tree that provides estimates using frequencies where weights are updated each step from root to target. On the contrary, NS employs logistic regression to filter out noise and updates weights that are important leading a more efficient time complexity.

2.3.3- Implementation and Hyperparameter Tuning

The W2V model will be implemented using the genism library. To begin with, experimentations emphasize on identifying key algorithmic variation such as the comparison of the SG and CBOW algorithms for optimal performance on the training set. Thereafter, the HS will be compared with the NS method using various sample sizes of the training data. After having selected an adequate approach, extensive experimentations will be conducted on various hyperparameters. Initially, experimentations will be carried out individually in order to determine suitable ranges of values for the following hyperparameters:

Sample → Allows us to set a threshold to designate the down sampling of high frequency words. According the genism documentation ranges between 10e-4 and 0 can be useful.
Minimum Count → A threshold to discard words with lower frequencies.
Window → Set a maximum distance between current and predicted sequence.
Alpha Values → Set an initial learning rate for the model.

Using the most suitable parameters filtered, various combinations will aim to determine how they behave together and find those optimally performing on the model. Finally, the vector size will be increased from the default value of 100 to 300. This value is consistent with word embedding language model experimented by Zweig and Burges (2011) and is regarded as an optimal value that yield adequate results.

2.4- Evaluation Methods

2.4.1 Accuracy

Among numerous methods used to evaluate the performance of a classifiers include accuracy, precision, recall and F1 score which are subsets of the confusion matrix. Although a confusion matrix would typically be an evaluation method of choice, answers (A-D) are not associated with each other in the testing data. Therefore, the accuracy metric will be used to measure the performance of classifiers and help identify optimal hyperparameter.

2.4.2 Perplexity

Perplexity is a metric used in n-gram models and is the inverse probability of test set normalized by the number of words. It can help identify the amount of meaningful information attained from the language model (Jurafsky and Martin, 2021). It is constructed without any knowledge of the test data and can only be compare language models if they use the same vocabulary. Although perplexity cannot concretely measure how the model will score on the test data, it can be a handy tool to analyze the scalability of the language model as whole rather than limiting the evaluation solely one testing sample. Therefore, perplexity will also be used as metric to draw comparisons on the n-gram models constructed.

2.4.3 Uncertainty Index

In this report an index has been devised to measure how uncertain a given predictions may be. This approach will reliant on the probability scores given for all the choices available for a given question. Respectively, choices with second highest probabilities will be divided by the choices with the highest probabilities. If both choices score a similar probability the uncertainty index will be closer to 1 (or 1 if both choices have identical probabilities). On the other hand, if the first choice scores a probability much higher than the second choice the uncertainty index will closer to 0 (or 0 if the second choice is 0). Using the uncertainty metric, a threshold value can be set for each model to determine the optimal scenario where accuracies scores are highest given this threshold. This may provide an avenue to combine various models to increase the quality of the solution. Furthermore, it

can also help investigate the types of questions each model is good at answering or not so good at answering by filtering out question using this metric.

3. Results

3.1 – N-gram Models (including baseline)

Perplexity Scores (Known: 2, Doc Size: 100)			
Unigram	**Bigram**	**Trigram**	**Quadgram**
642.1	30.78	19.72	14.3

Fig. 1

As a starting point, the perplexity scores were computed for each n-gram model [fig. 1]. The perplexity score for the unigram model is noticeably greater than all other n-gram models suggesting a greater level of ambiguity. On the other hand, all other n-gram model scored more similarly with the quadgram model having the lowest perplexity suggesting a lesser extent of ambiguity. The perplexity results attained suggest that predictions are less ambiguous given the increased sized of the context window.

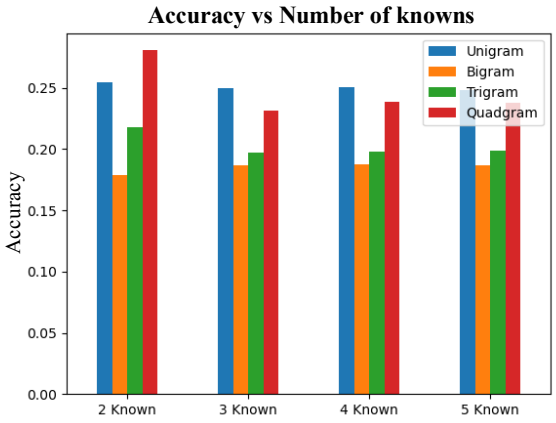

Fig. 2

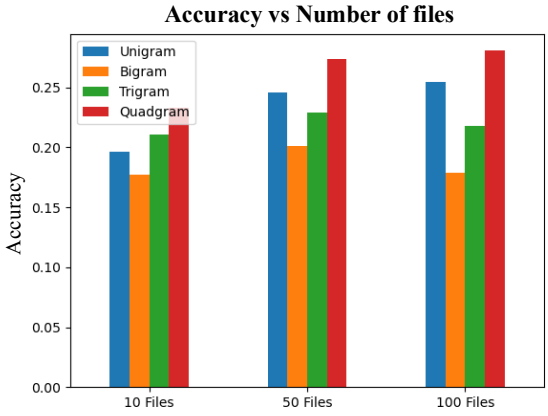

Fig. 3

All model scored their best accuracies with number of knowns set at 2 [fig. 2]. Respective the highest accuracy score attained was on the quadgram model as using 100 files to train the model which resonates with the quadgram perplexity score [fig. 3].. However, the unigram model scored a higher accuracy score than both the bigram and trigram models despite having a perplexity score approximately 20 times larger than the bigram and 32 times larger than the trigram model. This may be due to the fact that only a small sample of the training data was used meaning that a less target words are captured. Though one may question why the quadgram model was able score higher despite this assumption. Furthermore, the bigram and trigram models scored higher accuracy scores on 50 files compared to 100 files. Nevertheless, it is noteworthy that the corpus file consists of over 500 files which are randomly split suggesting potential disparities within the datasets.

3.2 – Word2Vec Model

As discussed in the methods section, initial investigations will focus on analyzing the algorithmic variation for the W2V model. Respectively, the CBOW and the SK algorithm were tested with both the hierarchical SoftMax as well as negative sampling using various sizes of the dataset. Document sizes of 10, 50, 100 and 200 were used to carry out this experiment [fig. 4]. Accuracy score were noticeably higher with larger sample of training data in for all the algorithms tested. Furthermore, it is also apparent

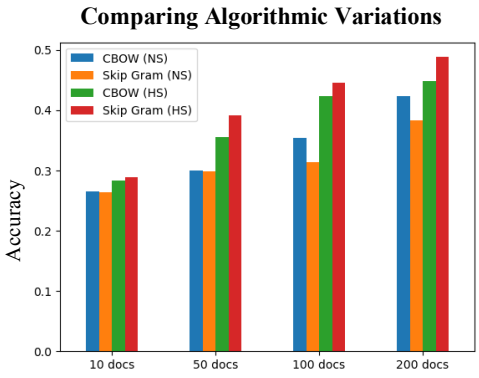

Fig. 4

that all algorithms maintain consistency for all sample sizes with CBOW an NS always outperforming.

Min Count	Accuracy	Window Size	Accuracy
2	0.49326	6	0.48942
7	0.48942	4	0.48750
5	0.48846	10	0.48653
4	0.48557	8	0.48557
3	0.48461	16	0.48557
6	0.48173	14	0.48461
8	0.48173	12	0.47884
1	0.46230	18	0.47884

Sample	Accuracy
0.01	0.49326
0.5	0.49038
0.1	0.48076
0	0.47788
0.001	0.47403
0.0001	0.47403
1.0	0.17788

Alpha	Accuracy
0.025	0.48365
0.05	0.48269
0.1	0.44807
0.01	0.41153
2	0.19903
1.5	0.19230

Fig. 5

The minimum count parameter sets a threshold effectively ignoring infrequent words. Similar to the experiment carried out for the n-gram models (frequency thresholding), the W2V model appears to have the highest accuracy score with a threshold of 2 [fig. 5]. Although most minimum threshold appear to have similar accuracies, a window size of 1 has a relatively lower accuracy score. Despite experimenting various window sizes, variations observed do not appear to very significant. Nonetheless, a window of 6 appears to be have the best performance followed by window of 4 which is not too far off. On the other hand, the sample size and alpha values see much greater variation. In this regard, sample sizes between 0.01 and 0.5 were found to have the best results [fig. 5]. Meanwhile, alpha values between 0.25 and 0.5 significantly outperformed all other values [fig. 5]. In this respect, further experimentations were carried out to further tune these hyperparameters. In doing so, optimal values of 2 and 6 were implemented for the minimum count and window size respectively and combined with ranges identified for the sample size (0.01, 0.1 and 0.5) and alpha value (0.025 and 0.05). Respectively, a sample size of 0.1 along with an alpha value of 0.025 we found to be the optimal parameters yielding an accuracy score of 49.52%.

3.2 – Comparison of Predictions

Test Model	Bigram	Trigram	Quadgram	Word2Vec
Unigram	12.5%	26.8%	31.7%	57.35%

Test Model	Unigram	Trigram	Quadgram	Word2Vec
Bigram	17.7%	27.3%	20.3%	39%

Test Model	Unigram	Bigram	Quadgram	Word2Vec
Trigram	31.3%	22.5%	65.2%	39%

Test Model	Unigram	Bigram	Trigram	Word2Vec
Quadgram	28.8%	13%	50.7%	52.39%

Test Model	Unigram	Bigram	Trigram	Quadgram
Word2Vec	30.2%	14.5%	14.5%	30.4%

Fig. 6

The tables above [fig. 6] exhibit the proportion of correct classification that each model has in common. The baseline models did not appear to have many predictions in common. Accordingly, the unigram shares 12.5% of its correct prediction with the bigram and 17.7% vice versa. This suggests that the baseline models behave in a distinct manner with a disparate proportion of correct predictions shared amongst them. On the other hand, the trigram and quadgram models share 31.3% and 28% with the unigram and 22.5% and 13% with the bigram respectively. This suggest that the bigram model can make more distinct prediction in comparison the other models. In contrast, data shows that the trigram and quadgram models share the highest portion of predictions in common at 65.2% which is consistent with the fact that their context windows consists of more than one word. Meanwhile the W2V model has by far the highest accuracy compared to all other model. Therefore, a combined

approach could see the bigram and quadgram models combined with the W2V model for optimal predictions.

In this respect further investigation will be carried out using the uncertainty metric to identify the potential combinations these methods aiming to enhance the quality of the solution.

Bigram

	Number of Sentences	Uncertainty Index	Accuracy
0	100	0.472166	0.242424
1	200	0.614284	0.216080
2	300	0.706830	0.210702
3	400	0.760938	0.197995
4	500	0.801885	0.194389
5	600	0.847906	0.198664
6	700	0.883909	0.195994
7	800	0.926786	0.193992
8	900	0.964095	0.184650

Quadgram

	Number of Sentences	Uncertainty Index	Accuracy
0	100	0.015372	0.575758
1	200	0.055660	0.477387
2	300	0.119059	0.431438
3	400	0.202390	0.393484
4	500	0.308049	0.348697
5	600	0.417805	0.327212
6	700	0.520409	0.310443
7	800	0.678730	0.309136
8	900	0.815369	0.295884

Word2Vec

	Number of Sentences	Uncertainty Index	Accuracy
0	100	0.698977	0.767677
1	200	0.764817	0.748744
2	300	0.815355	0.705686
3	400	0.848658	0.671679
4	500	0.884879	0.627255
5	600	0.910954	0.587646
6	700	0.935091	0.567954
7	800	0.958778	0.530663
8	900	0.977310	0.512792

Fig. 7

The tables above [fig. 7] illustrate how the models perform given using the sorted values of their uncertainty index over the test data. Although the performance of the bigram model appears to be constant, accuracy scores of 24.2% (index = 0.47) are recorded for the top 100 predictions with lowest uncertainty index. On the other hand, greater variations are observed for the quadgram and W2V models. While the quadgram model has an accuracy of 57.6% for the top 100 word with lowest uncertainty index. Meanwhile, the W2V model has an accuracy of 76.8% (top 100 words) and maintains an adequate accuracy 62.7% (index = 0.84) for the top 500 word with the lowest uncertainty index.

Considering these findings, these thresholds will be implemented to measure a hypothetical scenario where the three models are combined. If the W2V index is below 0.91 it will be selected as the prediction. Subsequently, if the quadgram index is below 0.15, it will return the prediction. Thereafter, if the bigram prediction has an index below 0.47 it will compute the prediction. Considering quite a few zero probabilities were assigned in W2V model an extra condition was used to revert the classification to the quadgram model. Respectively, if W2V prediction is not equal to zero, the quadgram prediction will used. Given non of these conditions are not met, W2V prediction will be used considering it's the best performing model. Result show that this approach can score an accuracy of 50.7 with the bigram model included. Finally, accuracy scores were increased to 51.8% following the removal of the bigram model.

4. Conclusion

This paper aimed to compare scaled n-gram and W2V model with a simple uni/bi-gram baseline implementation. Analysis on the n-gram predictions attained, highlights that they significantly vary despite sharing similarities in terms of their structures. On the other hand, great similarities were observed with the trigram, quadgram model and W2V model. It was found that when the context window size is increased in the n-gram models, it is able to capture wider contexts of each sentence and hence share more similar predictions with the W2V model. Finally, an adequate metric was devised to identify uncertainties in each model which was used to investigate a hypothetical scenario where these models may be combined. The result attained appear to be promising, leaving room for further investigation to this approach. Although, n-gram results were inferior to the baseline smoothed quadgram results attained by Zweig and Burges (2011), it is noteworthy that a small sample of training data was used and a combined n-gram approach was not implemented. Nevertheless, W2V model outperformed the baseline LSA used in by Zweig and Burges (2011).

5. Future Work

One key draw back of the n-gram models can be seen as it its inability to deal with unseen or infrequent words. One way of solving this problem is replace such words with synonyms. For instance, instead of creating an unknow word index, one could simply use the W2V model to extract a list of similar words effectively replacing ambiguity with words that may provide some contextual value.

In this paper, we presented a method to measure uncertainty. In doing so, our hypothesis shows that the model was able to attain better results using uncertainty index. This approach can be scaled to many other language models given that the predictions are made using probability. Among many, the BERT language model is known to be one of the best performing models available. Our experimentations suggested that various models behave differently in response to the test data. In this respect, the uncertainty index can be a useful technique to combine numerous model or even variation on models (with different hyper parameters) to further increase the quality of prediction. Nevertheless, methods used are highly corpus dependent making it hard to build a model that can deal with ever evolution of language over time.

6. References

Huang, P. S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013, October). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 2333-2338).

Mitra, S., Mitra, R., Maity, S. K., Riedl, M., Biemann, C., Goyal, P., & Mukherjee, A. (2015). An automatic approach to identify word sense changes in text media across timescales. *Natural Language Engineering*, *21*(5), 773-798.

Verma, G., & Srinivasan, B. V. (2019). A lexical, syntactic, and semantic perspective for understanding style in text. *arXiv preprint arXiv:1909.08349*.

Zweig, G., & Burges, C. J. (2011). *The microsoft research sentence completion challenge*. Microsoft Research Technical Report MSR-TR-2011-129.

Woods, A. (2016, August). Exploiting linguistic features for sentence completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 438-442).

Jurafsky, D., & Martin, J. H. (2021). N-gram language models. *Speech and language processing*, *23*.

Church, K. W., & Gale, W. A. (1991). A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech & Language*, *5*(1), 19-54

Kneser, R., & Ney, H. (1995, May). Improved backing-off for m-gram language modeling. In *1995 international conference on acoustics, speech, and signal processing* (Vol. 1, pp. 181-184). IEEE.

Naili, M., Chaibi, A. H., & Ghezala, H. H. B. (2017). Comparative study of word embedding methods in topic segmentation. *Procedia computer science*, *112*, 340-349.