Microsoft Research Sentence completion Challenge

Coursework Part A
Advance Natural Language Processing
Candidate Number - 251043

1.Introduction

The task involved building and evaluating different models for the Microsoft Research (MSR) Sentence Completion Challenge (SCC) [1]. Throughout the paper, and on its implementations, the focus is on comparing the models and exploring their differences and the effect that they have on performance.

1.1Sentence Completion Challenge

The authors of the MSR SCC noted that there were few publicly accessible datasets for semantic modelling. By addressing this gap, the authors provide a method for assessing a system's ability to model semantic meaning in text. The challenge consists of 1040 sentences derived from five of the original Sherlock Holmes novels. A low-frequency focus word was selected in each sentence and four alternatives were generated using a maximum entropy n-gram model. Here the challenge is to choose the original word, not any of the impostors. The n-gram model was trained on approximately 540 texts from the Project Gutenberg collection [1], with most selections being 19th-century novels. The word generation process adhered to various criteria, but all selected candidate impostor words had similar occurrence statistics as the original word. A human judge hand-picked each of the four impostor sentences from a larger set, whose job it was to pick sentences in which the generated word fit most naturally without obscuring the original correct answer.

1.2 Motivation

Implementing and evaluating the performance of various models on these test sentences means we can more easily examine the theory underpinning those implementations and identify nuances in the dataset. By documenting the changes in performance of the models, whilst attempting to maximise each one's score on the SCC, we can explore the effects of different features and hyper-parameter settings during the development process, and hope to gain insights into the way in which the different models process natural language.

2.Methods

The benchmark results for six distinct techniques are presented by MCR SCC [1]. Their human baseline was the highest scoring outcome. Using latent semantic analysis (LSA) similarity calculations of the cosines of angles between the vector forms of different words against the candidate word, the second highest scoring overall and highest computational baseline was achieved, which correctly answered 49 percent of the challenge questions. The remaining four approaches were all n-gram models in some form. All of these n-gram models scored in the 31 percent to 39 percent range.

This study looks at two distinct sentence completion systems and tests the features and parameter settings to determine how they affect accuracy. An n-gram language model composed of unigram, bigram, trigram, and 4-gram statistics is the first to be explored. The second method examines methods based on word embedding. The accuracy of the ngram and word embedding algorithms improves over time on the test sentences, and they can sometimes compensate for one another's flaws. For example, embedding models can give global semantic coherence while overcoming the local information constraint imposed on n grammes [2]. In addition, an ensemble

approach is tested, which aggregates forecasts in a voting system.

2.1 N-Gram Language Model

N-grams employ the conditional probabilities of previous words to approximate the probability of a word appearing in a phrase, i.e. they are statistical models that can estimate word probabilities based on context. The chain rule of probabilities, Markov assumptions, and maximum likelihood estimates are all used in this process. These three elements are related to estimating the probabilities of word sequences, the assumption that a word's likelihood is purely based on previously encountered words, and the fact that probabilities may be approximated by normalising n-gram counts, as seen in the corpus. N-gram models have shown to be effective in a variety of applications, including speech recognition and machine translation, as well as augmentative and alternative communication systems [3, 4]. These tasks are similar to sentence completion in that they all require assigning probabilities to word sequences. This was taken into account while deciding whether or not a ngram model would be suitable for this task.

2.2 Word Embeddings

Model Word embeddings are created by modelling words as points in a high-dimensional space, and Latent Semantic Analysis is the process of employing these dense vectors to represent semantics. [5]. Word2vec [6], which uses a neural network as a prediction model, and GloVe [7], which is a count-based method that uses co-occurrence information, have both contributed to this field. Words can then be compared by determining a measure between their vectors, usually using a dot product function like cosine or euclidean distance.This article looks at two different methods: Word2Vec and its fastText extension [8, 9]. For these models, intrinsic evaluation methods exist that compare the model's word similarity scores to those assigned by humans, such as the WordSim-353 [10] or the TOEFL dataset [11], which requires the model to identify the proper synonym for a target word. However, extrinsic evaluations for vector models are generally more useful because they allow one to see immediately whether there is any improvement in task performance [3], so evaluation for the embedding models will be based on the MSR-SCC challenge sentences and the model's accuracy on the test set.

3 Experimental Results

The findings of experiments with the n-gram and word embedding models, as well as an ensemble technique combining the two, are presented in this section. If models are trained on text (rather than being pre-trained), this will refer to a subset of the Project Gutenberg texts; more information on these, as well as some example test phrases for clarity, can be found in the model.

3.1 N-Gram Language Model

N-Gram	Perplexity		
	8	20	40
Unigram	107.0178	116.2954	221.4313
Bigram	51.6554	56.5050	74.3852
Trigram	37.5881	25.3478	17.0038
4-gram	12.3239	9.2456	6.4820

Tabel 1: Perplexity scores with increasing sizes of document the model trained on.

The first way of scoring takes an individual n-gram order, such as bigram, and returns the candidate token that maximises the candidate token's and context's bigram probability. Out of vocabulary (OOV) words are handled by supplying a numerical parameter - known - to the model at startup, which replaces all n-grams with occurrence statistics less than the known value with an unknown token.This n-gram model was initially trained on a growing number of Project Gutenberg texts, with perplexity calculations performed on a test set that was 20% the size of the training set. Because of decoding issues in some of the text files, any values given relating to the quantity of texts processed can be assumed to be estimated for expositional clarity. Table 1 shows how increasing the size of the training and testing sets affects perplexity for various

n-gram models; the perplexity test set is around 20% of the total number in the table. For comparison, the average sentence length, including words and punctuation, was 12.04, with an average of 7511 sentences per document.Sentences in the Project Gutenberg files relate to lines of text.

Table 1 indicates that as the n-gram size increases, the model fits the data better; this is a tendency that can be seen across all recorded document sizes. Perplexity comparisons between models trained on different volumes of text are impossible since they would not have used the same vocabulary [12]. The apparent coherence of the sentences formed using the n-gram models trained on 8 and 40 texts is similarly increased when visualised. The leftmost value corresponds to the size of the n-gram model employed, as seen below:

Document Size: 8

1. -the the , , the the the , ,

2. - and at the same time , and the.

3. - alone knew where Anne was.

4. - ’they shut or opened their gates with a trembling hand ,

Document Size: 40

1. - the,the.

2. - the same as the first.

3. - careless people should think.

4. - had spoken its simple reason through the lips of Dejah Thoris ’ prison before long.

In all three sizes, the unigram and bigram models are predisposed to choose the most common word sequences or words detected in the training data - stopwords and punctuation. The word token occurrence statistics for the model trained on 40 texts are shown in Figure 1. Figure 1 also explains the Shannon visualisation [13] sentences' short sentence length; the unknown token,' UNK,' is the most regularly occurring token and was chosen as a cut off for any sentence formed.Figure 2 investigates this possibility in the unigram and bigram scenarios by examining the effects of the known parameter on the perplexity of a model's test set. Because the training and testing data remained the same, the models below had fixed vocabularies.

Perplexity decreases as the number of known tokens increases, and more n-grams have their probability mass redistributed towards the unknown token (see Figure 2). When looking at the size of the vocabulary for models with different sizes of the known parameter, the largest reduction was from known = 1 to known = 6, which reduced the set size from 36971 to 9955. Table 2 displays the extrinsic evaluation: accuracy scores on challenge test phrases for models trained on 8, 20, 40, and 200 texts with no pre-processing or smoothing performed.

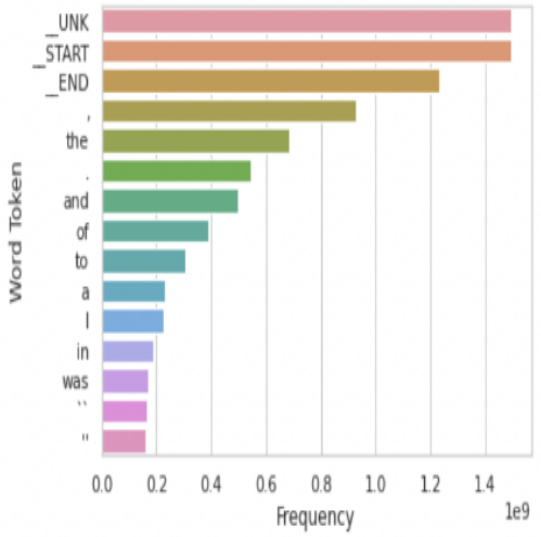

Figure 1: Word Token Occurrence Statistics.

For each, the known parameter was reduced to five. The scoring method considers the context of each candidate word, calculates the likelihood of that n-gram containing the candidate word occurring, and chooses the word contained in the highest yielding n-gram as the sentence's choice. This technique performs little better than random chance (20%) in the trigram and 4-gram situations, but shows some improvement in the unigram and bigram cases. The Asterisk next to the 19.90 value here and in subsequent experiments denotes that the n-gram model didn't have any information to help answer the challenge

sentence, thus it chose option A for that sentence by default.

The best scoring bigram model tries to use the context to the right of the target word rather than the left. The accuracy dropped from 27.60 to 21.54 as a result of this. Table 3 tries to explain this by looking at the most common word tokens on either side of the candidate word choices in the test sentences, as it was

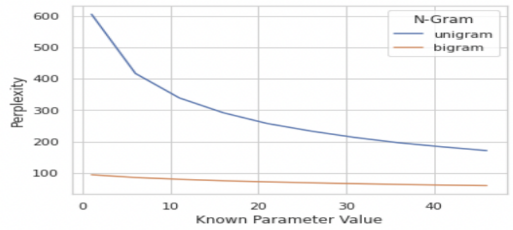

Figure 2: Perplexity against varying values of the known parameter.

hypothesised that a higher frequency of lexically inconsequential terms in the collection of right context tokens would be the cause.

N-Gram	Accuracy (%)			
	8	20	40	200
Unigram	25.87	25.29	23.75	24.90
Bigram	25.29	24.90	25.29	27.60
Trigram	18.85	19.13	19.13	19.33
4-gram	20.19	20.29	19.90*	20.77

Tabel 2: MSR-SCC score of n-gram models with varying training set.

A more accurate count of each occurrence of a stopword or a punctuation character revealed a higher quantity of these less meaningful tokens on the candidate choice's right side. Figure 3 shows updated word occurrence data for a model trained on 40 texts, with stopwords and punctuation removed and known reduced down to equal two. Due to their high frequency, the sentence start and end markers have been deleted from this image for clarity; the full image may be found in Appendix as Figure 8. The average sentence length was reduced from 12.04 to 5.79 tokens after stop words and punctuation were removed.

The target word in the MSR SCC test sentences can be placed in a context that exclusively contains stop words or punctuation.

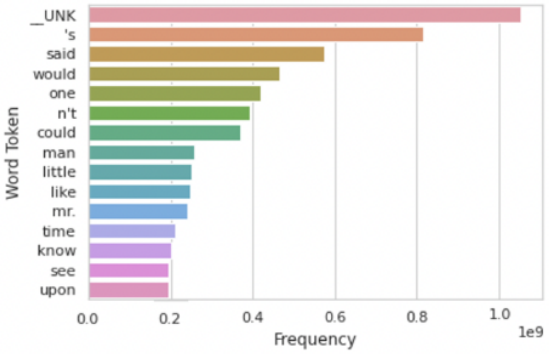

Figure 3: Word token occurrence statistics and punctuation removed.

All contexts are padded with the least number of unknown tokens the model may require to predict the correct candidate word for the phrase, ensuring that the extrinsic evaluation runs without errors. Table 4 indicates small improvements in the bigram situations, with a model trained on 200 documents attaining 30.1 percent accuracy, using the same technique of evaluation as Table 2 and with known set to 2.

Left	Right
the	,
a	of
was	and
to	in
and	the
had	to

Tabel 3: Most Frequent token used in left and right context bigram cases.

The trigram model fails until the training set size is expanded, whereas the 4-gram model never produces a match.

N-Gram	Accuracy (%)			
	8	20	40	200
Unigram	26.63	24.42	24.90	23.65
Bigram	21.73	24.04	27.21	30.10
Trigram	19.90*	20.19	20.29	20.67
4-Gram	19.90*	19.90*	19.90*	19.90*

Tabel 4: MSR-SCC score of N-gram models with stopword and punctuation removed.

After then, the "Stupid Backoff" approach [14] was put to the test. Backoff is a technique for coping with out of vocabulary (OOV) word sequences that uses available contextual information to estimate likelihood when higher order n-grams are unavailable. In addition to addressing OOVs, utilising less context can aid the language model by allowing it to generalise to less common circumstances [3].

Stupid Backoff does not generate normalised probabilities, instead relying on the n-grams' relative frequencies [14]. This frequency score distribution is described by the function S:

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{f(w_{i-k+1}^{i}}{f(w_{i-k+1}^{i-1}} & \text{if} f(w_{i-k+1}^{i}) > 0 \\ \alpha S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

The word is referred to as the backoff factor, and it serves as a context-independent weighting for each n-gram calculation order. The original paper's empirical evidence justifies using = 0.4. It was also discovered that utilising several values depending on the sequence of the n-grams can marginally improve the results [14]. The results of applying Stupid Backoff on n-gram models trained on 80 texts with known = 5 are shown in Table 5.

N-Gram	Accuracy (%)		
	Stop	Stop + Lemma	None
Unigram	24.81	24.42	25.29
Bigram	29.42	29.04	26.92
Trigram	27.12	26.54	27.12
4-Gram	25.19	24.23	27.5

MSR-SCC stupid Backoff score of N-gram model with and without stopword removal and lemmatozation (lemma) preprocessing (Tabel 5).

In Table 5, each n-gram row corresponds to the greatest rank of n-gram employed - for example, trigram. From trigram probabilities, Stupid Backoff will start backing off. Pre-processing the phrases appears to have little effect, and the 4-gram model performs about as well as the trigram when it is set to not penalise trigram weights, showing that 4-gram information is still scarce. Word tokens were lemmatized in the hopes of helping the model generalise better; nonetheless, it underperformed the version with simply stopwords (and punctuation) removed for each rank of n-gram.

Table 6 reveals that using the same scoring system as the MSR SCC basic 4-gram baseline [2] and the same bigram, trigram, and 4-gram probabilities, the results are more accurate. This approach matches the target word in n-grams up to 4-grams of the test sentence. Each n-gram match is given a different value (+1 bigram, +2 trigram, +3 4-gram, etc.), and the candidate word with the highest score is chosen. By increasing the training set size to a maximum of approximately 250 Project Gutenberg texts, the simple 4-gram evaluation method yields 34.9 percent correct, building on the best score given in Table 6 - maintaining known = 5 and not applying any pre-processing to the training or test data - the simple 4-gram evaluation method yields 34.9 percent correct.Stopwords were deleted from the same texts, yielding a 34.42 percent correct result.

N-Gram	Accuracy (%)		
	Stop	Stop + Lemma	None
Simple 4-Gram	31.15	30.87	31.25

Table 6 shows the MSR-SCC "simple 4-gram" score with and without sentence pre-processing.

3.2 Word Embeddings Model First

For Word2Vec and fastText, we tested pretrained embeddings (with 300 dimensions). Appendix A contains the base datasets for these models. Table 7 shows the results of their accuracy on the challenge sentences using vector similarity and distance metrics before and after sentence pre-processing. A complete similarity system [2] was used to get the scores in Table 7. This means choosing the candidate word that has the greatest average resemblance to all other words in the test phrase in the case of cosine similarity. This means choosing the candidate word with the shortest average

distance to all other words in the Euclidean distance case. The improvement can be seen by removing stop words and punctuation that have no lexical significance.

Embedding Method	Accuracy (%)	
	Cosine	Euclidean
Word2Vec	36.06	29.52
Word2Vec - PP	38.75	31.06
fastText	35.10	24.52
fasttext - PP	42.88	28.17

Table 7: Cosine and Euclidean total similarity test sentence with and without stopword removal (pre-processing).

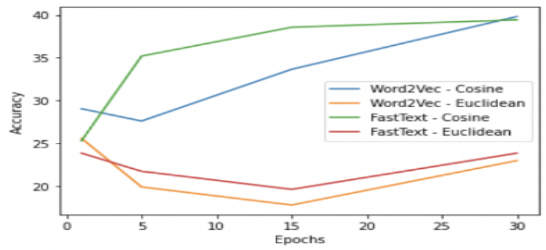

Figure 4: Accuracy of model on test sentences with a varying of epochs trained for.

However, because these findings are still below the LSA baseline provided in the original MSR SCC publication, embeddings are created using Project Gutenberg training data. Using these embeddings means more latitude in the hyper-parameter choices when developing the models, in addition to matching the training and testing domains (19th century books). Going forward, the same sentence preprocessing as in Table 7 is used.

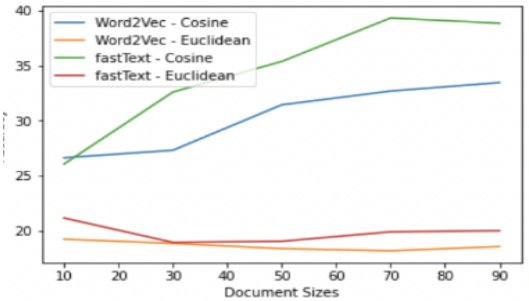

Figure 5: Accuracy of model on test sentence with increasing training document size.

In Figure 4, we investigate the impact of the epoch parameter on Word2Vec and fastText. The number of iterations the algorithm (skip-gram) performs over the corpus is determined by this. The models in Figure 4 were trained on 30 texts, with the rest of the parameters left alone (seeAppendix A). Unmentioned parameters are assumed to be set to default settings unless otherwise stated. After 15 epochs of training, both models' Euclidean distance measures drop down; nevertheless, after that, all metrics show a continuous increase. Figure 5 shows the consequences of increasing the amount of documents the model is trained on while keeping epoch = 15. The model's performance improves with the size of the training set in the case of cosine similarity, whereas Euclidean distance measurements underperform.Within a sentence, the greatest distance between the present and algorithm's predicted word is measured by window size.

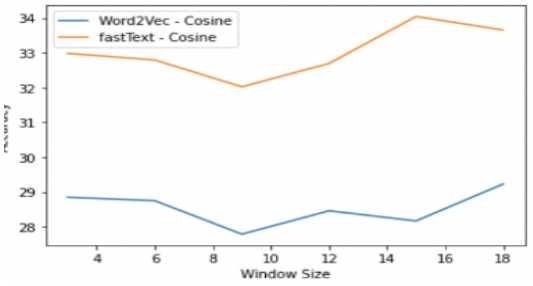

Figure 6: Accuracy of models with varying window sizes.

Figure 7 is obtained by keeping the parameters from Figure 6 but reducing the target word embedding dimensions from 300 to 100 and re-running the above window size test - this improves accuracy over the previous tests and reduces training time by a big factor.

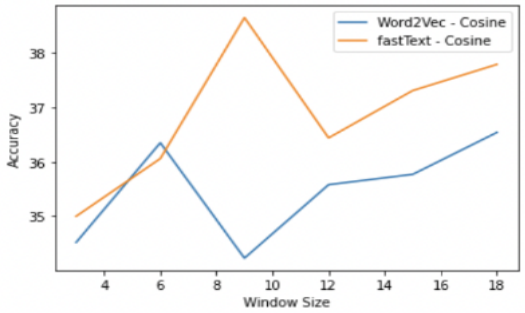

Figure 7: Accuracy of model of 100 dimensions with varying window sizes.

3.3 Ensemble Methods

In machine-learning systems, methods can be coupled to achieve even better outcomes [2, 15]. This ensemble approach was used to combine the scores from the two top performing models overall, pretrained fasttext embeddings and the simple4-gram, by normalising each score against the sum of the two sets of scores. The candidate word that received the highest overall score was chosen. This ensemble voting system scored 44.6 percent right, which was slightly higher than the individual scores.

4 Analysis

As the hyperparameters and data pretreatment procedures of all primary approaches studied were tailored to the task at hand, improvements were noted. Table 8 provides a summary of the top-performing models. The simple-4-gram model is the only one listed that did not use pre-processing and outperformed a similar model that did. Although the difference was small (less than 1% difference in accuracy), it's possible that stopwords provide useful context for the target word, such as inferring the required portion of speech for the candidate choice.

Model	Accuracy (%)
Simple-4-gram	34.90
Pretrained: Word2vec	38.75
Custom: fastText	39.33
Pretrained: fastText	42.88
Ensemble (fastText + 4-gram)	44.60

Tabel 8 : Best performing Model.

Figure 2 also showed that too high a known setting could potentially skip the model's learning process by over-generalizing its predictions for text with excessive probability mass stored for the unknown token. N-gram models are also sensitive to OOV words, so success was dependent on how the unknowns were handled via the known parameter. Table 1 shows that raising the size of n reduces perplexity, but the effect diminishes with each order of n, implying that there is a cut-off point where adding greater n-gram complexity does not translate into an equal effect on the models' ability to predict text.The trigram and 4-gram models initially performed worse than their unigram and bigram counterparts, but the addition of Stupid Backoff allowed the language model to utilise all available information, including the rarer trigram and 4-gram information. Because of the lower number of unique n-grams and the inability to fully preserve context after lemmatization, lemmatizing the text and deleting stopwords did not perform as well as just removing stopwords. The preprocessed simple-4-gram model predicted the target word as [laughing] when the correct response was [caught] in one of the tokenized pre-processed test phrases - ['tortured', 'tried', 'get', 'away','', 'tortured']. It's possible that the n-gram model learned more unigram, bigram, and even trigram versions with ['tried,"get,'away,'] and [laughing] than it did with [caught]. Although [laughing] is clearly out of place, this does not demonstrate semantic knowledge, but the n-gram model is at the mercy of its training data.

The pretrained fastText model received the highest challenge score of all the word embedding methods used. The cosine similarity metrics and Euclidean distance measures of both pre-trained Word2Vec and pre-trained fastText were enhanced by deleting lexically inconsequential terms. It was discovered that raising the value for the epoch parameter had a beneficial influence on accuracy on the test sentences for the custom word embeddings built from Project Gutenberg texts. With a fixed size of 30 training materials, Figure 4 depicted this relationship. When compared to the negative impacts of a smaller training set, a higher epoch value can counterbalance the bad effects of a larger training set, as seen in Figure 5.The Euclidean distance metrics' results for Word2Vec and fastText fell progressively as document size rose. This could be owing to a rise in the magnitude of vectorized words and n-grams, which would influence distance measures; cosine similarity measures, on the other hand, are unaffected because they only consider the angle between the vectors.

Cutting the dimensionality of the vector forms of words yielded improved accuracy while also drastically reducing training time for each bespoke model. This is in accordance with studies that suggest that a big dimension

setting can make the model harder and slower to train if there isn't enough data to learn [8]. For all window sizes examined in Figures 6 and 7, both fastText and Word2Vec models outperformed their dimensions = 300 counterparts at dimensions = 100, which is the suggested default for fastText.

Word	Window = 2	Window = 25
butcher	mastiff	baker
chicken	sweetbread	partridges

Tabel 9: Most similar word at different window sizes.

Smaller window sizes gather information about the current word and any words that appear nearby, whereas larger window sizes capture information about the topic [16]. Table 9 shows this with Word2Vec models that were trained on 30 documents with various window sizes. This exemplifies one of the ways that word embedding methods outperform n-gram models, which are limited to accessing only local information due to their context sizes. Another improvement can be noted with fastText in particular. By aggregating the n-grams that make up the OOV word, fastText can represent OOV occurrences by computing word embeddings using the vectors of substrings of characters contained inside the wordIn this scenario, pre-processing may have broken the right answer into distinct tokens (assuming tissue-paper was present in the training data), forcing the simple-4-gram model to use the unknown token for [tissue paper], but fastText was able to aggregate the n-grams to generate a correct prediction. The ensemble method, which combined the normalised candidate word scores of the fastText and simple-4-gram models into an equally weighted voting mechanism, improved both methods' solo challenge accuracy, although it only delivered 18 correct responses to phrases that neither model got right separately. The ensemble technique favoured the embedding method's prediction 132 times over the n-gram prediction and the simple-4-gram method's prediction 110 times over the word embedding prediction in terms of correct answer types. When the simple-4-gram and word embedding approaches agreed, the remaining correct answers (204) were given.

5 Discussion

The MSR SCC can be passed using n-gram and word embedding-based two method implementations, according to this paper. The perplexity reported from a test text can be reduced by striking a balance between the size of the training set and the n-gram model's strategy for dealing with unknowns, potentially indicating an improvement in semantic coherence. As a result, the accuracy of the challenge questions has improved. Different orders of n-gram can give information to advance this using Stupid Backoff and the simple-4-gram approach. In general, word embedding models that use total cosine similarity outperform n-gram models, and they can have settings tailored to the training set to increase accuracy even more.The results of the ensemble approach were subjected to a one-tailed binomial hypothesis test to establish the unlikelihood that their performance was attributable to random chance; the test resulted in the null hypothesis (that the model was performing as random chance) being rejected. Appendix A has the complete results. Further research into the effects of popular smoothing methods on n-gram models, as well as the effects of the known parameter as the training set size grows, could be profitable. Similarly, rather than exclusively testing skip-gram on domain-specific word embedding methods, researching the effects of employing Word2Vec and fastText continuous bag of words algorithms would have been an intriguing topic to investigate.

6 Conclusion

The strategies for sentence completion are investigated in this report. The two methods based on n-grams and word embedding give a good benchmark for the challenge, as they can accurately predict the proper word for a sentence using computationally simple and affordable methods. Both strategies outperform random chance, with the word embedding methods getting slightly less than half of the answers accurate. Finding more effective ways to incorporate local and global sentence information in ensemble algorithms to obtain even improved accuracy would be an extension of this work.

Bibliography:-

[1] C. Shannon, “Prediction and entropy of printed english,” Bell System Technical Journal, vol. 30, pp. 50–64, 1951.

[2] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, “Large language models in machine translation,” 2007.

[3] D. Jurafsky and J. H. Martin, Speech and Language Processing (2nd Edition). USA: Prentice-Hall, Inc., 2009.

[4] K. Trnka, D. Yarrington, J. McCaw, K. F. McCoy, and C. Pennington, “The effects of word prediction on communication rate for aac,” in Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, ser. NAACLShort ’07. USA: Association for Computational Linguistics, 2007, p. 173–176.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” Journal of the American Society for Information Science, vol. 41, no. 6, pp. 391–407, 1990.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in ICLR, 2013.

[7]T. Landauer and S. Dumais, “A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.” Psychological Review, vol. 104, pp. 211–240, 1997. J.

[8] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, “Advances in pre-training distributed word representations,” in Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” CoRR, vol. abs/1607.04606, 2016. [Online]. Available: http://arxiv.org/abs/1607.04606.

[10] “Placing search in context: The concept revisited,” ACM Trans. Inf. Syst., vol. 20, no. 1, p. 116–131, Jan. 2002. [Online]. Available: https://doi.org/10.1145/ 503104.503110.

[11] J.Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.

[12] C. Buck, K. Heafield, and B. V. Ooyen, “N-gram counts and language models from the common crawl,” in LREC, 2014.

[13] G. Zweig, J. Platt, C. Meek, C. Burges, A. Yessenalina, and Q. Liu, “Computational approaches to sentence completion,” 50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference, vol. 1, 07 2012

[14] G. Zweig, J. Platt, C. Meek, C. Burges, A. Yessenalina, and Q. Liu, “Computational approaches to sentence completion,” 50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference, vol. 1, 07 2012.

[15] T. G. Dietterich, “Ensemble methods in machine learning,” in Proceedings of the First International Workshop on Multiple Classifier Systems, ser. MCS ’00. Berlin, Heidelberg: Springer-Verlag, 2000, p. 1–15.

[16] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 302–308. [Online]. Available: https://www.aclweb.org/anthology/P14-2050 8.

[17]https://github.com/LordLean/Microsoft-Research-Sentence-Completion-Challenge/blob/main.

A.Appendix:-

Example Sentences from the MSR SCC:

Available at https://www.microsoft.com/en-us/research/project/msr-sentence-completion-challenge/ Two example sentences are given with their five candidate word choices listed below. The original and correct answer is in bold and the string ‘ ’ marks the target position of the word in the sentence.

I have it from the same source that you are both an orphan and a bachelor and are alone in london.
The resident matched walking, came out of the room and closed the door tightly behind him.panther guard country-dance.Image displaying the occurrence statistics of n-gram model trained on 40 texts, known = 2. This image includes crying instantaneously the start and end sentence markers: ” START”, ” END”

Figure 8: Stopwords and punctuation were removed from the word token occurrence statistics. Word Embeddings that have been pre-trained: Licensing and other information are available at: https://github.com/RaRe-Technologies/gensim-data

fasttext-wiki-news-subwords-30 Wikipedia 2017, UMBC web based corpus and statmt.org news dataset (16B tokens)

word2vec-google-news-300 Google News (about 100 billion words)

Default parameters for the custom embedding models can be found at:

Word2vec https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec

For a one-tailed test, a Binomial Hypothesis Test with a confidence level of 95% (= 0.05) was used. There were a total of 1040 trials, with our ensemble model completing 464 of them successfully. Trials (questions) are considered to be mutually independent, with the probability of a given outcome being the same for all. A correct response has a one in five probability of being chosen, whereas an incorrect answer has a four in five chance of being chosen.

fastText https://radimrehurek.com/gensim/models/fasttext.html#gensim.models.fasttext.FastText.

Let x equal the number of times the model answers a question successfully. Let π equal the probability of success in any one trial.

H0 : $\pi \leq 0.2$ i.e. due to random chance. H1 :$\pi > 0.2$.

$P(x \geq 464) = 7.26e{-}72$ and so we reject the null hypothesis at the 5% significance value because the returned p-value is less than the critical value of 0.05.