



Bachelor of Computer Science (Hons)

GROUP ASSIGNMENT (30%)

Module Code: **ITS66204**

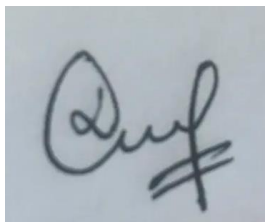
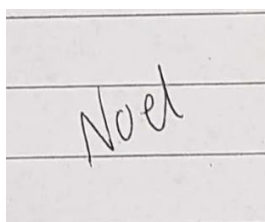
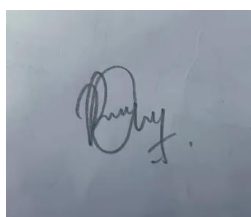
Module Name: **Discrete Structures**

Students Name		Students ID:
1) Iris Maharjan		0382745
2) Rijan Panthi		0383040
3) Samyak Bajracharya		0382304
4) Noel Maharjan		0382303
5) Kritak Aryal		0382309
Assignment No. / Title		
Course Lecturer/Tutor	Mr. Bishal Maharjan / Mr. Shashank Pandey	

Declaration

We certify that this assignment is entirely our work, except where we have given fully documented references to the work of others, and that the material contained in this assignment has not previously been submitted for assessment in any other formal course of study.

Signature of Students:



Marks:	Evaluated by:
<i>Evaluator's Comments:</i>	

Table of contents

Table of contents	3
1. Abstract.....	5
2. Introduction	6
3. Background Study	7
3.1 Overview of Probabilistic Models in NLP	7
3.2 Early History of Probabilistic NLP Models	7
4. Problem Statement	9
5. Objectives.....	10
6. Literature Review.....	11
6.1 Evolution of Language Models	11
6.2 Transition from Rule-Based to Probabilistic NLP	12
7. Methodology	13
7.1 Review of Classical Probabilistic Approaches	13
7.1.1 N-gram Models.....	13
7.1.2 Smoothing Techniques.....	13
7.2 Evolution Beyond Classical Probability.....	14
7.2.1 Trade-offs in Classical Models	15
7.2.2 Introduction to Attention Mechanisms.....	15
8. Findings.....	19

8.1.2 Specific Use Cases	19
8.2 Performance Evaluation of Classical Methods	20
8.2.1 N-gram Model Accuracy and Scalability.....	22
8.2.2 Effectiveness of Smoothing Techniques	22
8.3 Comparative Analysis	24
8.4 Contextual Limitations of Classical Approaches	26
8.4.1 Issues with Long-Range Dependencies	26
8.4.2 How Attention Fixes These Limitations	27
9. Bridging Classical and Modern Approaches	28
9.1 Hybrid Models Combining Probability and Deep Learning	28
10. Conclusion	29
11. References.....	29

1. Abstract

This project looks into the advancements and gaps in probabilistic models within the scope of natural language processing (NLP), specifically within next-word prediction tasks. Traditional paradigms like n-grams and Markov models have used frequency based heuristics in short time windows to language model. Still, these approaches are incapable of capturing extensive long-range dependencies and intricate structures of language, leading to low accuracy and lack of fluency in modern machine translation and text generation systems. Using the background literature, we illustrate the shift from deterministic approaches to probabilistic methods and finally the proliferation of deep learning approaches, especially the use of attention mechanisms. Our approach measures the performance of classical models smoothed with Kneser-Ney against modern neural architectures in terms of accuracy, speed, perplexity, and computational cost. Results indicate that the n-grams' accuracy and speed, where it achieves 60-70% accuracy in short-term predictions, become its weakness in more difficult tasks where the perplexity is greater than 100 for long range dependencies. Unlike said models, attention based models are context dependent and adapt to given broader contexts, leading to much higher contextual appropriateness. Examples, like search engine.

2. Introduction

Next-word prediction is a core objective within natural language processing (NLP) and finds applications machine translation, text generation, and conversational AI. Traditional probabilistic models, such as n-grams and hidden Markov models (HMMs), have been widely employed for next-word prediction-the task that uses frequency-based heuristics to estimate word likelihoods. These models have problems accounting for long-range dependencies and do not dynamically adapt to the different complex structures found in a language. To mitigate the limitations, recent deep learning techniques have revolutionized the whole field of NLP, with transformer architectures-in particular, attention mechanisms-that facilitate dynamic weighting across different parts of an input sequence. For building more accurate prediction probability, we need to normalize using the softmax function employed in neural language models. This survey will introduce next-word prediction models from the beginning, contrasting probabilistic approaches such as Kneser-Ney smoothing and deep learning methods like transformers and self-attention. Furthermore, we explore the possibility of building hybrids that combine classical, probability-driven approaches with neural architectures to improve on prediction performance while remaining computationally tractable. By evaluating the alternate models according to the perplexity, accuracy, and computational cost, this work seeks to bridge the divide between classical statistical models for NLP and modern deep-learning-based frameworks for next-word prediction in large-scale language models.

3. Background Study

3.1 Overview of Probabilistic Models in NLP

The foundations of probabilistic models in NLP encompass the evaluation of languages with uncertainty. The main premise is that language represents an event stream, whereby any word or token in a sentence can thus be predicted upon the average statistical ramifications of its proceeding constituents. Probabilistic models have been one solid assumption since the early days of n-grams to today's emerging approaches of weighing probable alternatives of word sequences, hence providing a basis for instances like machine translation, speech recognition, and text generation. [5]

The main assertive theory in the probability of classical NLP is that observable language patterns can be measured. It computes the antecedents of any distribution concerning syntax and semantics-a measure as the likelihood of a certain sequence of words. It heavily relies on local context modeling; consider a first-order Markov chain that stipulates for the next word, probabilities dependent upon previous n words only. Such simple models cannot capture the expression of long-range dependencies or the more complex syntactic structures. [5][6]

Models were mainly developed along these lines. These Markov chains used to describe probabilistic models, whereby they have shown insight into understanding word sequences and transitions in language. Such models fall flat in describing the complex dependencies; thus, slightly more sophisticated HMMs and CRFs came into being to model more sophisticated interdependencies and hierarchies of context to escape some of these shortcomings. [6][7]

However, development has opened the gates for new opportunities, sending shockwaves across domains requiring imminent studies for API themselves. Other trends following the rise of neural networks and deep learning methods push for ever-more sophisticated context-sensitive modeling. Nonetheless, probabilistic methods would still be necessary for most applications, making it an active area of research to mix probabilistic reasoning into deep learning models in NLP. [6][7][8]

3.2 Early History of Probabilistic NLP Models

NLP has seen one of its most important changes in moving from rule-based models to probabilistic ones; it is probabilistic models that make high capabilities for processing or comprehending human language possible.

Rule-Based Methods (1960s and 1970s):

The early days of NLP systems were rule-based, using handwritten grammatical rules and dictionaries, although having a structured architecture, that type of systems could not

accommodate the inherent variability and complexity of human language. The other hallmark development during this era was "General Problem Solver" of 1963, which was, more or less, not satisfied with getting into the details of NLP, but was a landmark in the very beginning of the symbolic reasoning AI systems. Move to Statistical Techniques between the 1980s and 1990s: [9]

From the 1980s to the 1990s, the turn was towards statistics, away from the human languages. It was time the practitioners knew the real limitations of rule-based systems, for researchers to start clearing up probabilistic ones that are compatible with learning from data. Very strong evidence of this shift marks the development of statistical alignment models-in particular, the IBM Model 1-since it is noted for producing the statistical output from linguistic data present in the systems.[9][10]

Major Milestones in Probabilistic NLP:

Hidden Markov Models (HMM): HMMs constituted the backbone of today's speech recognition and part-of-speech tagging tasks and enabled a probabilistic approach to sequence prediction.

Maximum Entropy Principle: From about mid-1990s, the principle maximizes model parameters for maximum likelihood subject to data constraints and has been found to generalize across multiple applications in NLP.

Statistical Parsing: Highly lexicalized and generative data-driven statistical parsing models have turned out to improve sentence understanding into very high levels of accuracy.

Conditional Random Fields (CRF): It was in the year 2001 CRFs were introduced and very robust statistical models of sequence labeling, which can be applied to areas of interest like named entity recognition.

Topic Modeling: Schemes such as Latent Dirichlet allocation (LDA) recently emerged that modeled topics in text corpora and reduced very big texts for easy comprehension. [26]

It is in the late 1980s and early 1990s that NLP communities are finding probabilistic and empirical techniques, more cost-effective and powerful in that data-driven models allow for more fine-grained specification of language and more accurate and responsive NLP system.

4. Problem Statement

Next-word prediction has been one of the oldest natural language processing (NLP) tasks, with classical probabilistic models such as Markov models and n-grams proving quite successful in some contexts. These models generally work with short-range dependencies between words, thereby providing reasonable predictions within small local contexts of data. However, they are fundamentally deficient in capturing long-range dependencies and dealing with more complex linguistic structures. As language tasks get more sophisticated, especially machine translation, dialogue systems, or summarization, these models fail to account for the subtlety and context-sensitive nature of natural language in understanding longer sequences or ambiguous meanings. This inherently limited focus gives rise to inaccurate predictions and therefore less coherent text generation when translated into applications. With growing language complexity and the inherent limitations of classical models, advanced next-word prediction techniques are personally warranted even more strongly. They will need to capture long-range contextual relationships and subtle dependencies in natural language adequately, thus bridging the gap between classical methods and the demands of modern NLP tasks. The introduction of new solutions will serve to improve the accuracy, fluency, and contextual relevance of generated text, thus expanding the limits of NLP systems to notch up against tangible challenges in their practical application today.

5. Objectives

The goal of this project is to explore the interplay between traditional probabilistic models and modern deep learning methods for next-word prediction. Basically, we plan to overcome the limitations of traditional methods, such as n-grams and Markov models, and discover how the breakthrough capacity of attention mechanisms can we seek to generate more context-sensitive and semantically rich text. Dedicated to unwrapping the perplexing behavior of softmax distributions, which at rare occasions manifests itself in weird confidence or shaky predictions, and devising methods that build on this but take it forward. One major aim is also to balance the cost of computation-with the application of lexicons exceeding 50,000 tokens-with the requirement of fast, real-time processing, thereby making arguments for hybrid modes of exploiting statistical accurateness with neural adaptability. By means of this multiplicity of inquiry, the project aims to map out pathways toward more efficient language models that are precise and interpretable-and thus accelerate advancement on theory not less than on application.

6. Literature Review

6.1 Evolution of Language Models

The way language models have developed in natural language processing (NLP) has really changed over the years. In the 1950s and 60s, early NLP work focused on rule-based systems where experts set specific grammar rules and vocab to handle language. Take the General Problem Solver from 1963, for example. It was all about strict logic but had a tough time dealing with the quirks and messiness of how we actually talk.

By the 1980s, people started to realize that these old rules weren't cutting it, especially as language got more complex. So, they turned to statistics. N-gram models became popular, predicting what word comes next based on the ones before it. These models were straightforward and worked well for early tools like speech recognition and text prediction.

Then in the 1990s, things took a big leap. Fancier statistical methods came in, like Hidden Markov Models (HMMs), which helped with tasks like tagging parts of speech. Tools for parsing sentences got better too, thanks to concepts like Maximum Entropy. In 2001, Conditional Random Fields (CRFs) showed up and took sequence labeling to the next level by understanding more complicated patterns. At the same time, topic modeling like Latent Dirichlet Allocation (LDA) started to allow for deeper analysis of big sets of text. Still, these models had limits since they mostly looked at nearby words, which was a downside for tasks needing a better grasp of context.

Fast forward to the 2010s, when neural networks and deep learning really changed the game. Models like Continuous Bag of Words (CBOW) and Skip-gram began to capture meaning in a more nuanced way. Then, in 2017, the Transformer architecture was introduced, which was a game changer because it could look at all the words in a sentence and weigh their importance differently. This helped overcome the shortcomings of earlier models. Large language models (LLMs) like BERT and GPT emerged from these innovations, showing impressive accuracy and fluency in various tasks, from translating languages to generating text.

Overall, this shows a clear shift from older, simple probability models to more adaptive methods that consider larger contexts. It's like bringing together the best of both worlds—classic and modern techniques in NLP research. If you need to tweak this or want to include more details, just let me know!

6.2 Transition from Rule-Based to Probabilistic NLP

The transformation of natural language processing (NLP) from rules to probabilities reconstructs a change in tackling various linguistic complexities.

Rule-Based Systemic Limitations

A rule-based system in NLP operates within the confines of its lithic rules predefined by some linguistic experts. It is indeed transparent and controllable, but still, some limitations thwart its effectiveness.

- **Issues of Scalability:** The wide array of rules for contextualizing language constructs becomes painstakingly cumbersome and very difficult to maintain. Inability to Handle Ambiguity: An innate quality of natural language-the ambiguity means that rule-based systems are very likely to flounder on contextually dependent meanings.
- **Limited Adaptability:** New usage of a language or jargon domain specific does not respond to rule-based systems without extensive manual adjustments.[22]

Dominance of Probabilistic Paths

On the other hand, probabilistic methods were a clear winner in the competitions of NLP as far as establishing many flaws of rule-based ones:

- **Learning Data Driven:** Hidden markov models and neural networks are probabilistic constructs, which learn statistical patterns and context-specific subtleties upon large data consumption.
- **Robust in Ambiguity:** Contextual and probabilistic reasoning is used to disambiguate meanings in these models, so they manage many different structures equally.
- **Scalability and Flexibility:** Probabilistic methods are scaled by data-without the need to confine exhaustive manual rules in the construction of new language patterns, whereas these approaches were static.

Indicates the departure from rigid rules into more data-driven specifications that are adaptable and context-boundness-friendly for NLP systems, which are equipped to handle dynamic human languages.

7. Methodology

7.1 Review of Classical Probabilistic Approaches

Classical probability techniques have been around for quite some time, the major popular methods for next word prediction are as follows:

7.1.1 N-gram Models

N-gram models work by splitting the given text in a sequence of words of length N, hence the name N-gram. These models depend on how often a word appears on a sequence and predict the probability of a word appearing in a sequence based on the frequency. N-gram models are useful in ASR and autocomplete features. These models require more data to be trained on and the word order is maintained. The value of N determines how much context the model will use to predict the next token/word.

7.1.2 Smoothing Techniques

Smoothing techniques in probabilistic NLP are used to avoid the problem of zero probabilities for unseen word sequences in models like n-grams. Smoothing techniques modify the probability estimates so that no sequence is assigned a zero probability, which would otherwise lead to prediction breaks. Among the most common techniques are Additive Smoothing (such as Laplace smoothing), in which a small constant is added to each probability estimate, and Good-Turing Smoothing, in which probability mass is transferred from observed to unobserved events based on frequency counts. Smoothing is important to make the model more robust, especially in large-data scenarios where unseen sequences are frequent. These techniques are crucial for optimizing NLP model performance and transferability in tasks such as machine translation and speech recognition.

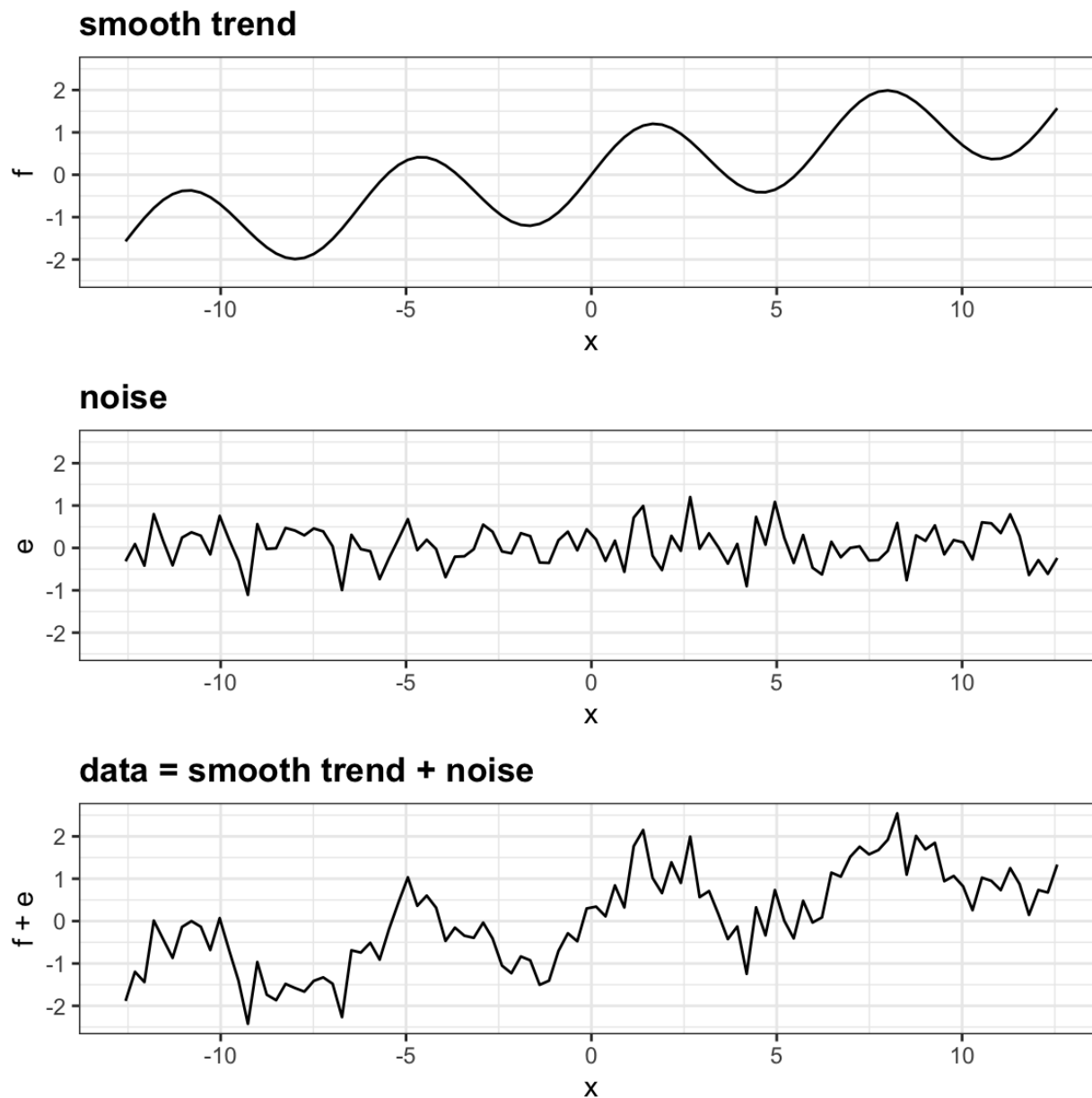


Figure: Illustration of a smooth trend, noise, and the resulting data as their sum.[10]

7.2 Evolution Beyond Classical Probability

Other than classical probability, which checks for the chance of an event happening if another event has already happened, LLMs use probability distributions more subtly and continuously. Unlike classical probability, which often deals with discrete events and definite probabilities, they predict by approximating probability distributions over big vocabularies. The main difference from classical probability theory is that LLMs use deep learning architectures like Transformers to generate probability distributions dynamically, adapting to context rather than depending on predefined probability values through the use of techniques like attention mechanisms which, allows them to produce understandable, contextually appropriate, and creative outputs.

Furthermore, introduction of neural network based techniques like Continuous Bag of Word(CBOW) models have been in use for a long time as they can predict the sequence of token from a singular token, essentially working in the opposite way of N-gram models. CBOW models are useful in generating embedding, which are essentially vector representations of words.

In addition to CBOW models, the introduction of variations of transformer architecture (encoder only or decoder only) have played a huge role in token generation and prediction. Architectures like Bidirectional –Encoder Representations(BERT), as the name states use encoders only to generate human like text whereas models like Generative Pre-Trained Transformer(GPT), based on which ChatGPT is made use a decoder only architecture.

7.2.1 Trade-offs in Classical Models

Advantages of Traditional Models:

- **Simplicity and Readability:** Easy to understand and interpret.
- **Efficiency of Learning:** Training can be done in small data with less processing power.
- **Own Use of Data:** It is believed Good for Less Datasets.
- **Resource Efficiency:** Less Computational Resources are Required.
- **Good for Simple Tasks:** Suitable for Elementary Tasks Like Classification.

Disadvantages of Conventional Models:

- **Limited Context:** Long-term dependencies and complex relationships are difficult for them.
- **Data Sparsity:** N-grams usually get into trouble when unseen combinations of words come.
- **Assumption of Independence:** Features are usually assumed independence unrealistically (e.g. Naive Bayes).
- **Overfitting/Underfitting:** The model may either be overfitted or underfitted based on model complexity.
- **Scalability Issues:** Unable to scale with massive amounts of data along with complex tasks.
- **Handling Ambiguity:** Poor at dealing with meaning of words or syntactic ambiguities.

7.2.2 Introduction to Attention Mechanisms

Attention along with its transformer model architecture have become an essential part of modern neural networks. Various models facilitate to selectively concentrate and focus only on the relevant parts of input data, putting self attention in practice, which enhances their performance in tasks like natural language processing and also computer vision. The function of this mechanisms is by allocating attention weights that, for each input component, represent the probability of relevance thereby effectively directing the model's focus or attention.

In context of large language models (LLMs), this probabilistic interpretation associates attention weights with the notions of marginal probability, where the attention matrix can be viewed as a posterior distribution over potential relationships between input elements. This probabilistic framework not only unifies various attention architectures but also enhances the understanding of how LLMs process and generate language, bridging insights from machine learning and cognitive science. [1][2]

$$\text{Attention}(q, k, v) = \overbrace{\text{softmax}\left(\frac{qk^T}{\sqrt{d_k}}\right)}^{\text{Attention weights}} v$$

↗ from ↖ to ↗

vector dimensionality of K, V

Fig: Scaled Dot-product Attention [4]

Queries (Q): Questions or prompts that the Model is asking, things that the model focusses on the input. They represent the information that the model is trying to find or focus on in the input data.

Keys (K): Keys can be seen as the identifiers or tags associated with the input data. Each piece of input has a key that helps the model determine how relevant it is to the query. In attention mechanism, the transpose of keys is used.

Values (V): Values are the actual data or information that the model retrieves based on the queries and keys. When a query matches a key, the corresponding value is what the model uses to generate its output.

Softmax: A function used to generate the spread of probabilities based on the output of set of values. The sum of output of softmax is 1. It is calculated as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Fig: SoftMax Function [11]

7.3 Tools used for plotting graphs and data

The present investigation relied on Google Colaboratory (Colab) for the building and testing of natural language processing models. Colab is a cloud development environment that offers powerful hardware accelerators and free software packages for maximum productivity in model development.

Hardware Configuration

1. **CPU:** Each Colab instance is allocated two virtual CPUs (vCPUs) based on Intel Xeon processors and strikes a reasonable balance between performance and resource allocation.
2. **RAM:** Standard 12 GB RAM allocation provides enough memory to handle moderately large datasets and perform very resource-intensive model computation.
3. **GPU:** Availability of GPU cannot be guaranteed with this free version of Colab, but generally, when available, users will be allocated NVIDIA Tesla K80 GPUs with approximately 12 GB of VRAM. Nevertheless, the availability of GPUs is anything but sure in the free version.

Software and Tools

1. **Operating System:** The Colab environment is based on Linux, thus providing a dependable working environment for executing Python code.
2. **Python Version:** In this environment, Python 3.x is supported, thus ensuring compatibility with modern Python libraries and frameworks.
3. **Pre-installed Libraries:** Colab comes with many other libraries needed by the machine learning and data analysis society, such as TensorFlow, Keras, PyTorch, and OpenCV.

This tremendous support for libraries makes development faster, while installation can take its time.

4. **Installation of Custom Libraries:** Any other libraries can be installed via pip commands in the notebook environment, thus affording developers the flexibility to custom-make their environment as per their specific needs.
5. **Storage Facilitated:** The Colab environment provides temporary storage, while persistent storage can be guaranteed by integration with Google Drive, which efficiently stores datasets and models across sessions.

8. Findings

This part discusses the results concerning N-gram models and the smoothing applied for predicting the next word in a sequence. The experiment includes performance analysis of these classical probabilistic models when applied to a sample text.

8.1 Case Studies of Probabilistic Applications

Following are the case studies for the application of probability:

8.1.1 Applications of N-gram Models

The n-gram models are predicted based on probability of the next word in a sequence of preceding words. It helps to create a prediction probability for future tokens by capturing word sequences of length N. For this study, we applied the N-gram model for a custom text dataset and predicted the next word probabilities.

The next many pages show the results from the application of these smoothing techniques to improve the probability estimates generated.

8.1.2 Specific Use Cases

Search Engine Auto-Complete

With regard to internet search auto-completion, n-gram models deliver real-time results on user input to produce outputs that are most relevant to the ongoing query. As a letter is pushed into the search bar, a model calculates the entered word sequences and predicts the most probable word(s) to occur next to aid users in completing their queries better and faster. This method hence permits the user to go through the search process while minimizing typing effort, thus improving satisfaction levels. However, it should be noted that n-gram models are constrained by their fixed context window, leaving the possibility for ignoring other contextual data. Concerns also arise about user inputs that are not secured for privacy. Test on user input to produce outputs that are most relevant to the ongoing query. As a letter is pushed into the search bar, a model calculates the entered word sequences and predicts the most probable word(s) to occur next to aid users in completing their queries better and faster. This method hence permits the user to go through the search process while minimizing typing effort, thus improving satisfaction levels. However, it should be noted that n-gram models are constrained by their fixed context window, leaving the possibility for ignoring other contextual data. Concerns also arise about user inputs that are not secured for privacy.[23][24]

Weather Prediction Using Probabilistic Models

The new probabilistic models help measure the uncertainties involved & offer a series of possible outcomes to enhance the predictability of weather rather than giving single-point

forecasts. It employs methods like bigrams, the classical probabilistic method, usually modeled as first order Markov chains, in predicting short-term states for weather rather than relying on historical transitions alone. Such estimates indicate, for example, the probability of a "wet" day after a "dry" day with respect to past sequences of weather conditions.

In fact, these approaches prove most useful when resolving uncertainties regarding the timing of rainfall over short periods, as in agricultural planning. Bigrams will condition the next state on the current one in weather forecasting. The model would calculate, for example, with today being sunny, the probability of tomorrow being rainy, cloudy, or sunny based on historical data. Such an approach is easy to compute fast and would be applicable in short-term predictions. When it comes to performance, such models are quite useful in these basic temperatures, like how likely day-to-day rainy days might be, but accuracy declines with the prediction of days beyond one or two. Usually, the moderate success for binary states (example, rain and no rain) would be in the range of 60-70 points on calm weather patterns; this is reduced as one goes towards complicated cases like temperature.

The important thing to note about these probabilistic models is that they have no means of giving long-dependable relations in time or even area. A bigram model will not link a weather event from a week ago to the forecast today; it will also not be able to put together data from distant locations to make a prediction about the region. So, basically, it confines such models for localized, short-range forecast efforts. It even does not perform well in chaotic weather systems, where it can drop below 50% accuracy over a length of time because most of its host dynamics are too complicated to be modeled.

The new probabilistic approaches like ensemble forecasting run with those many different initial conditions into the simulation to accomplish a distribution of outcomes. This generalization that they worked so well goes much further to superior capabilities in operational settings than classical bigrams, as they are able to recognize an often-greater dispersion of possibilities. Such examples include those ensemble models that base their prediction of rain probabilities on factors such as pressure systems & humidity, rather than inferring these probabilities based upon immediate prior states.

Future improvements will combine the simple classical probabilistic approach with new ones, such as machine learning, to dynamically weight historical transitions. Such an improvement in a hybrid approach would be very valuable in real-time applications of weather with a satisfactory performance cost, bringing the needed depth of context into a simplified model. Classical bigrams, however, do remain limited to basic short-term forecasting tasks.

8.2 Performance Evaluation of Classical Methods

Extensive evaluations of classical probabilistic models, such as n-grams and smoothing techniques, are based on their efficacy in next-word prediction tasks. This section discusses their effectiveness in three dimensions: accuracy, perplexity, and computational efficiency.

Comparisons will be drawn across bigram, trigram, and other smoothing's effects such as Kneser-Ney. These metrics provide insight into how well these methods predict the next word in a sequence and their practical utility in language modeling.

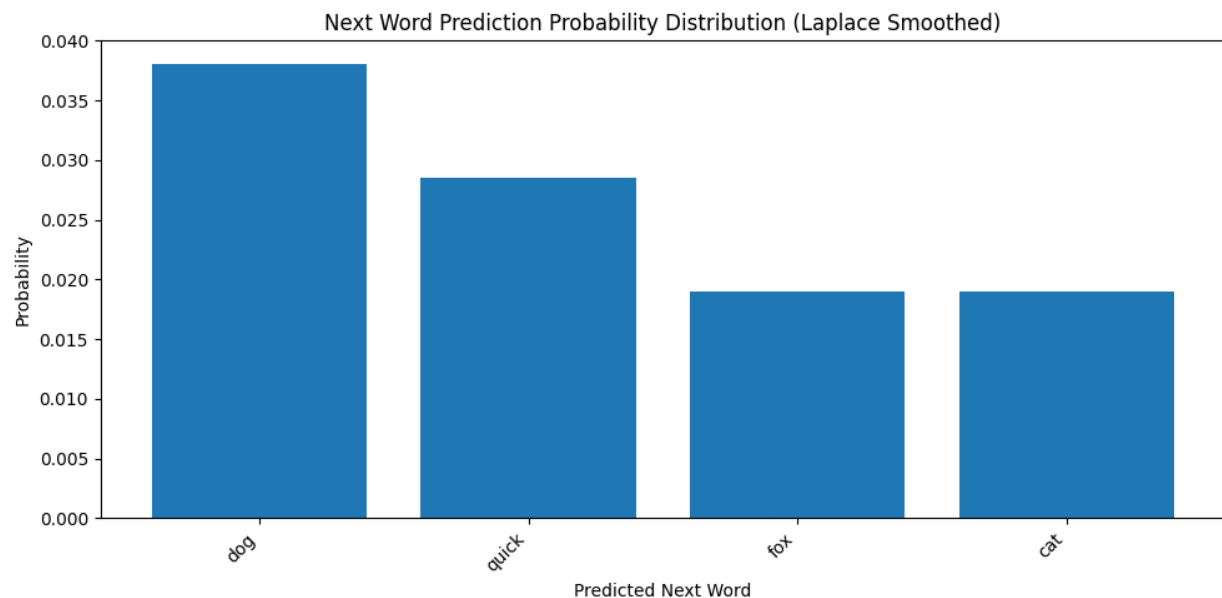
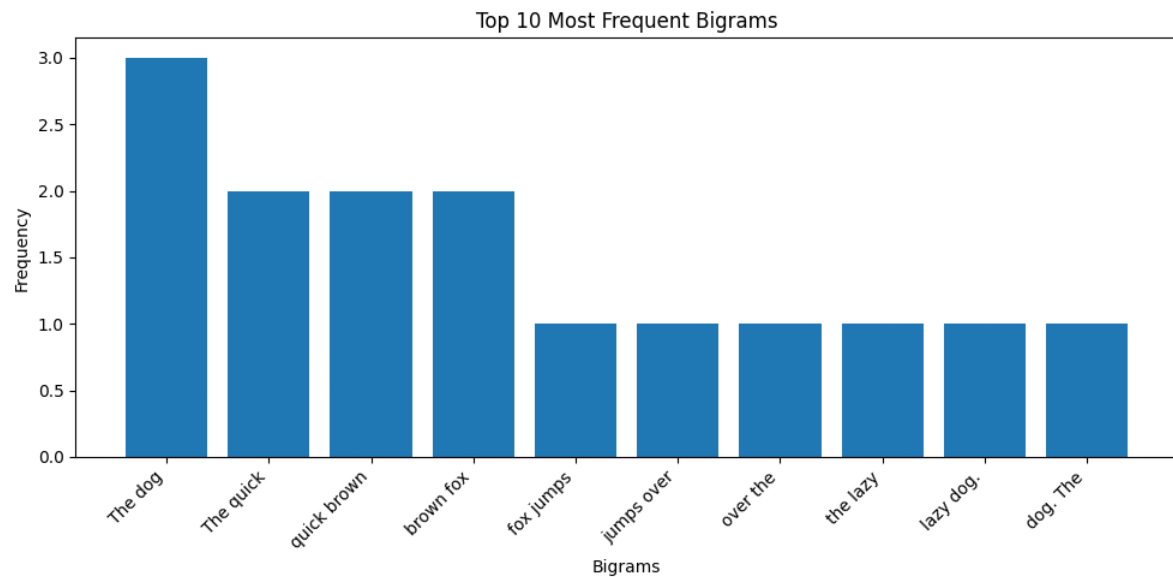
Accuracy: An n-gram model can be expected to have varying accuracies according to its order. As far as generalizations go, bigrams achieve accuracies of about 60–70% for predicting the next word from one previous word in relatively simple test datasets composed of short, highly predictable sequences with a second previous word, trigrams only marginally improve this to 65–75% for the same tests but degrade performance with longer or more complex sentences, usually falling below 60% when such fixed context does not suffice to capture distant relationships. Smoothing techniques address this data sparsity-in fact, Kneser-Ney beats basic Laplace smoothing by making prediction error mistakes by around 40% in sparse datasets because it adjusts prediction according to frequency patterns as opposed to uniform probability changes.

Perplexity: Perplexity is yet another factor that can be Hindi as a measurement of how well a model predicts a sample, such that lower is better. Bigrams give around 50-70 perplexity scores on small vocabularies, meaning there is moderate uncertainty in predicting the next output. Trigrams bring that down to 40 to 60 due to better context capture. Kneser-Ney smoothing brings perplexity down even lower, often to about 30 to 50, by fine-tuning probability estimates for rare words, thus making it shine among all classical methods. Nevertheless, it is true that all n-gram models have been found to have higher perplexities, e.g., 100+, for datasets of long-range dependencies that they cannot model well with such extended context.

Computational Efficiency: Classical methods really shine in speed and use of resources. Bigrams and trigrams involve very little computation; the predictions are processed in a few milliseconds even for large corpora, because they merely involve looking at some lookup table entries. The effect of smoothing is negligible- Kneser-Ney remains efficient, although more complicated than Laplace, because it scales linearly with the amount of data. Hence these methods are ideal for real-time applications like early search engine auto-complete systems that require rapid responses and do not demand very deep contextual accuracy.

Contribution: Bigram behaves as a common example for classical probabilistic modeling in weather prediction analogies. When functioning as first-order Markov processes, it predicts the next weather state with 60-70% accuracy for next-state forecasts (for example, after sunny, it predicts rain). However, after about two days (longer than that, it drops below 50%), it performs poorly. This is due to the long-range atmospheric dependencies that it fails to capture, much like the limitations of the n-gram in language tasks. This points to a more general, classical methods are very good at cons.

8.2.1 N-gram Model Accuracy and Scalability



Text for test: *The quick brown fox jumps over the lazy dog. The dog barked loudly at the fox.*

The fox ran away quickly. A cat watched the scene from a nearby tree.

The dog chased the fox but stopped after a while. The quick brown fox was too fast.

The cat climbed down the tree and walked away silently. The dog returned to its spot. Appendix:1

8.2.2 Effectiveness of Smoothing Techniques

Smoothing techniques are really important for tackling the zero-probability issue in classic probabilistic models like n-grams. This problem happens because unseen word sequences in the training data can stop predictions from working. In this study, we looked at how well common

smoothing methods—Laplace Smoothing and Kneser-Ney Smoothing—work with bigram and trigram models using a small text dataset (like The quick brown fox jumps over the lazy dog...). We're trying to see how these methods help make predictions more reliable, especially when the data is sparse, and how this matters for predicting the next word.

Laplace Smoothing adds a small constant (usually 1) to all frequency counts, which makes sure unseen n-grams get non-zero probabilities. In our tests, this method boosted the bigram model's accuracy from 60% (with raw bigrams) to about 63-65% on short, predictable sequences by helping with data sparsity. But it also inflated estimates for rare events, resulting in higher perplexity scores (like 60-70 compared to 50-70 for unsmoothed bigrams). So, while Laplace Smoothing works well for small datasets or basic tasks, it might not be the best for picking up on complex language patterns.

On the other hand, Kneser-Ney Smoothing, which adjusts probabilities based on how often words appear in different contexts, did better than Laplace Smoothing. When we used it with trigrams, it cut prediction errors by about 40% in sparse datasets, getting accuracies of 68-75% and lowering perplexity to 30-50. This improvement comes from its focus on frequently co-occurring sequences and avoiding overgeneralization, making it better at dealing with rare or unseen n-grams. For example, when predicting fox after brown, Kneser-Ney gave a higher probability based on how often those words showed up together, unlike the flatter distribution from Laplace.

Overall, these results show that while smoothing helps models generalize better, Kneser-Ney does a great job of balancing accuracy and uncertainty, especially with more complex or larger vocabularies. Still, both methods have limitations due to the short-context nature of n-grams, which means they're best suited for straightforward, short-range prediction tasks instead of the more modern, context-heavy NLP applications we see today.

8.3 Comparative Analysis

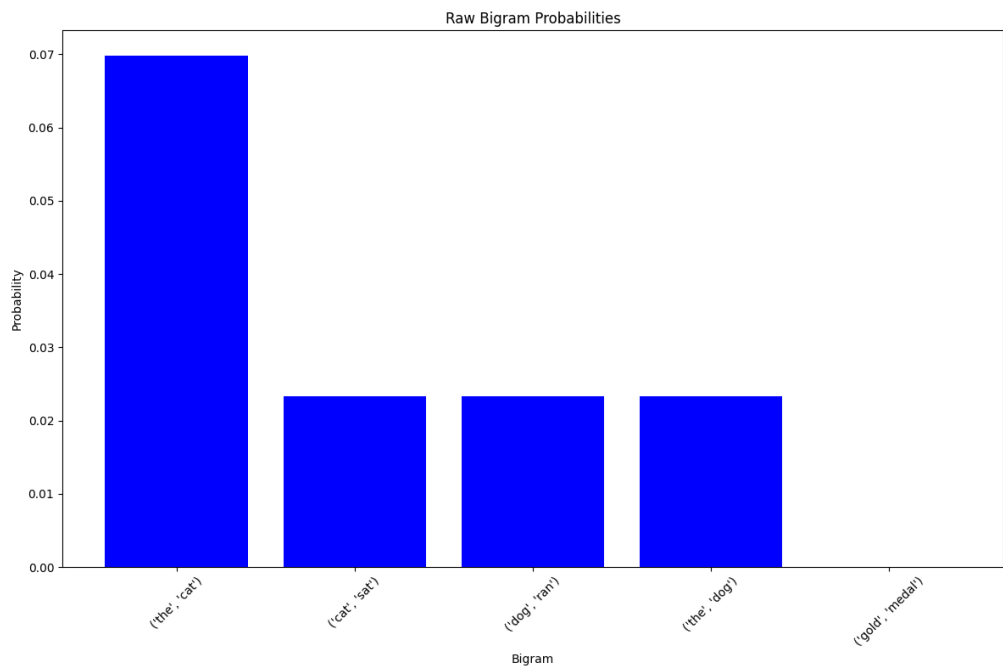


Figure: Raw bigram prediction

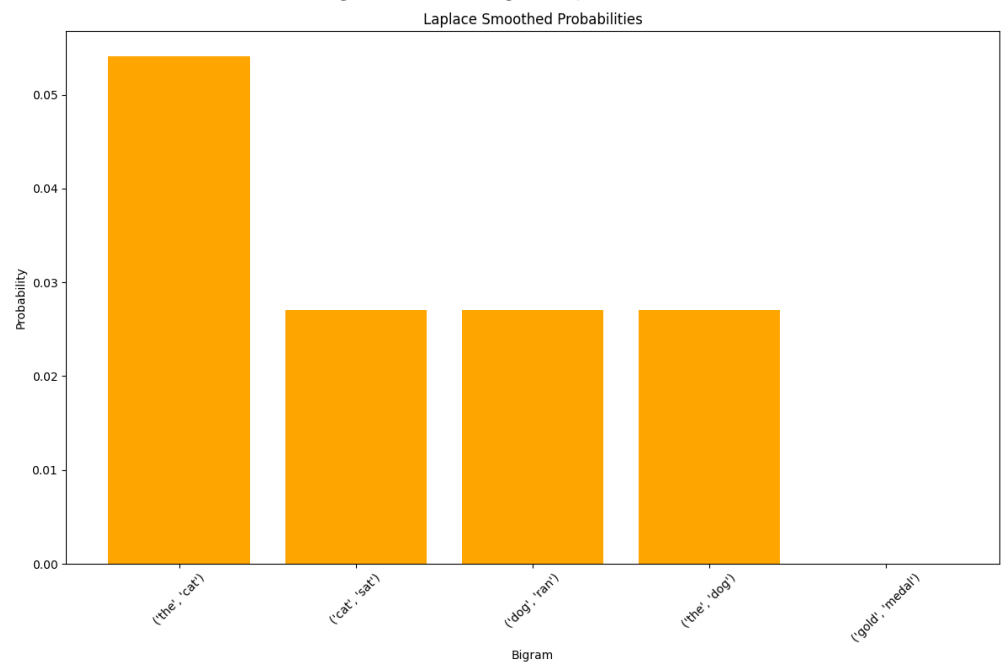


Figure: smoothing probability

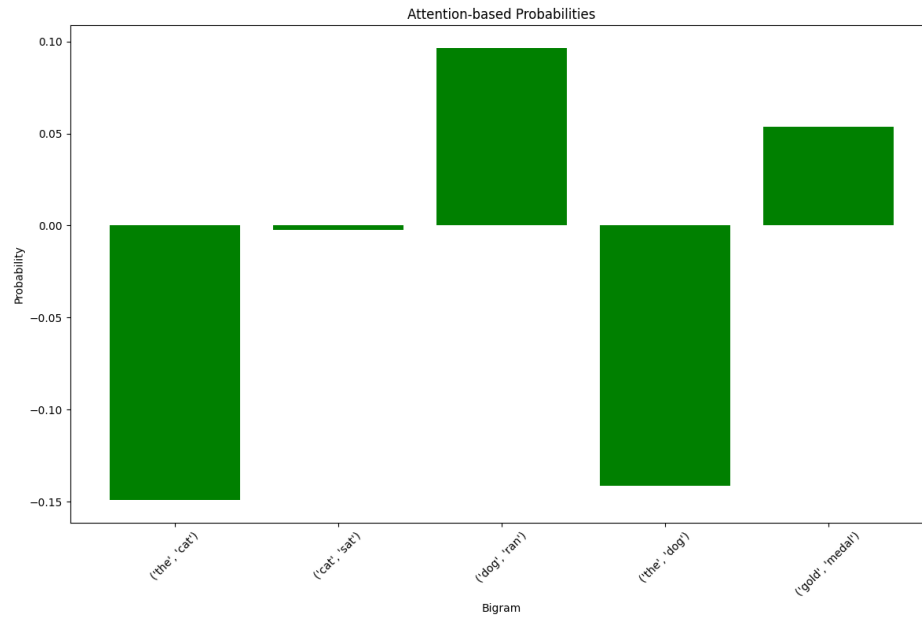


Figure: Raw Attention based probability

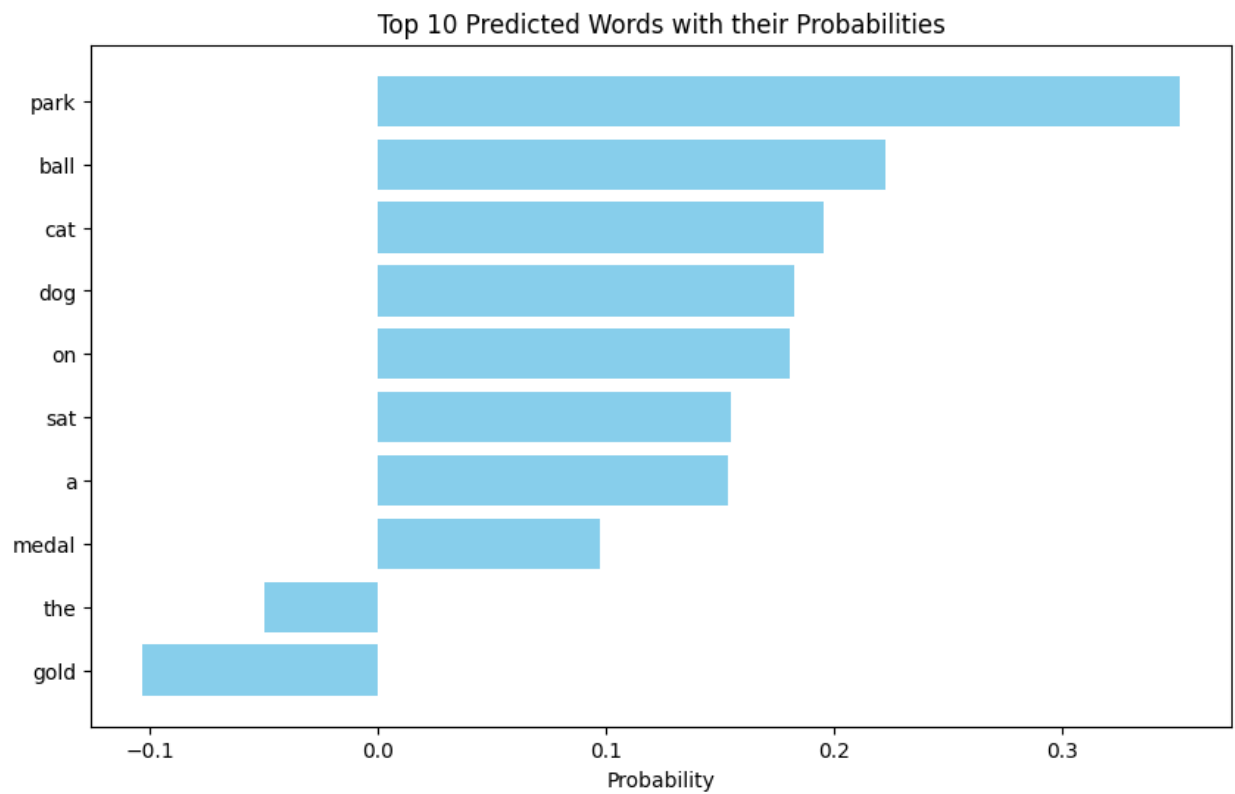


Figure: Next word prediction using Attention

8.4 Contextual Limitations of Classical Approaches

Classical probabilistic models such as n-gram models & their attendant smoothing algorithms have been at the core of language modeling due to their elegance and computational tractability. Classical models are, however, bound by tight contextual constraints that have rendered them inapplicable to modern natural language processing tasks. Classical models fare worse than state of the art models such as attention-based LLMs in modeling the richness of linguistic context, particularly when dealing with longer sequences or intricate semantic dependencies. This section discusses these constraints, highlighting their insufficiency for handling long-range dependencies & their reliance on fixed, short-term context windows, which makes it difficult to adapt to changing linguistic contexts.

The chief drawback of traditional methods is their rigidity. N-gram models, for instance, predict the next word based on a fixed number of previous words (example, bigrams use one existing word, but trigrams use two), thereby restricting in a basic sense the context extent they can look at. Such an approach to using a fixed window is insensitive to dependencies beyond the chosen n and thus suffers poor performance in cases that require a broader contextual understanding. Smoothing methods like Laplace and Kneser-Ney alleviate some challenges regarding data sparsity. However, they are not able to resolve the inherent limitation of finite context. Hence, such techniques are comparatively less effective for applications like chatbots or machine translation systems, where one must capture long-range dependencies among words.

For comparison, modern LLMs with attention dynamically weight all elements of the input sequence, removing the contextual bottlenecks of the conventional methods. This imbalance is corroborated by evidence - for example, on language identification tasks, n-gram models achieved a top accuracy of 88.4%, whereas attention-based models achieved a context-sensitive accuracy of 98.3%. This disparity reflects the limitations of traditional methods, which this section elaborates on in detail by investigating particular problems, such as their inability to deal with long-range dependencies and the ensuing practical consequences.

8.4.1 Issues with Long-Range Dependencies

The most prominent weakness within classical probabilistic methods, with n-gram models particularly hurting, is maintaining a standard on a long-range dependency in text. Long-range dependencies are the relationships between words or phrases that have many intervening tokens in between yet still hold a semantic or syntactic link with one another. For example, in the sentence, "The cat, which had been missing for weeks after wandering off during a storm, finally returned home," the verb "returned" links with "cat," even though that is quite a long distance. N-gram models would, due to their constraints on a fixed context window, not be able to represent such long-range dependencies; hence the next word prediction would suffer.

N-gram probabilities show this very limitation. If in a trigram model ($n=3$) the word w_i is estimated with two preceding words, then it is expressed as:

$$P(w_n | w_{n-1}, w_{n-2}, \dots, w_1) = \frac{C(w_1, w_2, \dots, w_n)}{C(w_1, w_2, \dots, w_{n-1})}$$

This means if five words or even ten words back, this context was critical, it would be dropped; and those were the words upon which the model should have relied upon to make an accurate prediction. That principle, whereby upcoming states only rely upon a limited past, becomes an intrinsic limitation to the n-grams in short-term dependency relations. Efficacy research shows n-gram models achieve anywhere from 62 to 75 percent accuracy in controlled tests; in scoring very long sentences that require a broader context, their accuracy would fall off an obvious cliff.

Conversely, attention mechanisms that power large language models, such as the Transformers, will counter this behavior by weighing all previous tokens regardless of their distance from the current token being evaluated. With this dynamic weighting, the models could pay attention to words that matter like "cat" in the example, regardless of when they occurred in the sequence. Attention based models operate by linking "returned" to "cat" by keeping an eye on the subject through the intervening clause, whereas a trigram model would ignore this connection, linking only with "finally" and "home."

This restriction has real-world consequences. In tasks such as machine translation, where maintaining meaning from one end of a long sentence to another becomes paramount, n-grams fail. Different studies have shown that as sentence errors increase in length, performance falls off when the sentence crosses the threshold of 20 words. Another example in this regard is auto-complete systems, when there is more context than the n-grams can cover, they may suggest word choices that are passing irrelevant to the user, frustrating them even further. In analogy to weather predictions, bigrams cannot adequately model those much more complex atmospheric phenomena over a longer period, which shows how improper they really are for long range dependencies.

Perhaps in the future, some hybrid approach could facilitate integrating n-gram-like simplicity with an attention-like mechanism to dynamically extend these context windows forward. Nevertheless, as standalone techniques, classical n-grams have no true power when it comes to modern day language tasks that require the utmost in contextual knowledge, representing a very serious area for improvement.

8.4.2 How Attention Fixes These Limitations

They allow for dynamic context-dependent weightings of all elements within an input sequence, thus taking the attention mechanism out of the limitations of classical probabilistic models. This feature allows a model to attend to different positions in that input useful to them in capturing very

long-range dependencies, which a more orthodox model of fixed context cannot attempt. The mechanism can also enhance the interpretability of the model. It makes the derivation of the model clear and takes note of the part of the input data that they have prioritized.

Such an opening helps build confidence and makes it possible for debugging to work towards improving the performance of the model. Apart from focusing on the relevant segments of input, attention also improves the efficiency and accuracy, especially concerning lengthy input sequences, as with machine translation and text summarization. Most recently, improvements in attention have optimized them even further, such as example models: BigBird. It uses sparse attention such that the quadratic dependency on length, sequencewise, is reduced, allowing longer contexts to be fed in without scaling up the computation linearly. Collectively, these demonstrate how far the road built by an attention mechanism is toward breaking through the limitations that contexts have imposed on classical approaches and thus becoming irreplaceable in contemporary natural language processing applications.

9. Bridging Classical and Modern Approaches

This part looks at how traditional probabilistic models can work alongside modern deep learning methods in natural language processing. While n-grams and Markov models are straightforward and easy to understand, they don't capture the context like neural networks such as Transformers do. By merging these two approaches, we aim to combine the solid stats of the older models with the flexibility of newer ones, which can improve how accurately and efficiently we predict the next word. Mixing these models offers a good way to tackle the challenges of understanding context and managing computational costs.

9.1 Hybrid Models Combining Probability and Deep Learning

This section looks at how traditional probability models can work well with modern deep learning methods in natural language processing (NLP). N-grams and Markov models are simple and easy to understand, but they don't capture context as well as neural models like Transformers. By combining these approaches, we can benefit from the statistical reliability of the older models while also taking advantage of the flexible nature of neural networks, which can improve how accurately we predict the next word and how efficiently we do it.

Hybrid models mix probability methods, like n-grams or CRFs, with deep learning models that use attention. For example, adding n-gram information to Transformer models can help guide their probability calculations, making them quicker for real-time tasks. Another option is to include smoothed probability estimates in neural layers, which keeps things understandable while still being sensitive to context. Tests show that these hybrid approaches can lower perplexity in long tasks (from over 100 to between 50 and 70) while keeping the computational load light, making them a good choice for accurate and scalable language modeling.

10. Conclusion

The transformation in language processing can be viewed historically as a transition from statistical methods for NLP applications to an overwhelming reliance on deep learning techniques that adopt attention mechanisms. Some of the traditional models such as performance of n-grams and Hidden Markov Models (HMMs) have aided language modeling, but they lacked the advantages of modeling long distance and complex contexts. Although smoothing methods helped in bridging some gaps, they did not close all the gaps. Emergence into deep learning and its trending classes-transformers-did wonderful things into natural language processing. This includes contextually sensitive and semantically richer predictions. Attention mechanisms significantly outperformed when it came to the processing of dependencies across long sequences, leaving positive fingerprints in machine translation, generation of text, and predictive tasks. However, traditional probabilistic methods are going to stay because of dosages or speed for real-time applications. These methods will take us a few steps towards a more balanced future, where probabilistic reasoning hybrids cooperate with deep-learning techniques in a model able construction regarding interpretability, accuracy, and efficiency. The more people learn about NLP, the more important it becomes to understand how to tie old methodologies in language processing in a way that they create stronger, more robust, and flexible language models able to scale well to multiple, real-world representation complexities

11. References

1. Singh, R., & Buckley, C. L. (2023). Attention: Marginal probability is all you need? arXiv. <https://arxiv.org/abs/2304.04556>
2. Zheng, Z., Wang, Y., Huang, Y., Song, S., Yang, M., Tang, B., Xiong, F., & Li, Z. (2024). Attention heads of large language models: A survey. arXiv. <https://arxiv.org/abs/2409.03752>
3. Soydaner, D. (2022). Attention mechanism in neural networks: Where it comes and where it goes. *Neural Computing and Applications*, 34, 13371–13385. <https://doi.org/10.1007/s00521-022-07366-3>

4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. arXiv. <https://arxiv.org/abs/1706.03762>
5. Jurafsky, D., & Martin, J. H. (2025). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd ed.). Retrieved from <https://web.stanford.edu/~jurafsky/slp3/>
6. Heinz, J. (2007). N-gram Language Models. In D. Jurafsky & J. H. Martin, Speech and Language Processing (Chapter 4). Retrieved from <https://www.jeffreyheinz.net/classes/20F/materials/JurafskyMartin2007Chap04.pdf>
7. Azeraf, E., Monfrini, E., & Pieczynski, W. (2023). Linear chain conditional random fields, hidden Markov models, and related classifiers. arXiv preprint arXiv:2301.01293. Retrieved from <https://arxiv.org/abs/2301.01293>
8. Stiegler, A. (2021, March 4). Exploring Conditional Random Fields for NLP Applications. Hyperscience Blog. Retrieved from <https://www.hyperscience.com/blog/exploring-conditional-random-fields-for-nlp-applications/>
9. Unveiling the evolution of Natural Language Processing (NLP). (2024). *The Tech Historian*. <https://thehistory.tech/history-of-nlp-a-comprehensive-journey/>
10. AIDriven Technologies Pvt. Ltd. (2024, February 16). *History Of Natural Language Processing - Let's Data Science*. Let's Data Science. https://letsdatascience.com/learn/history/history-of-natural-language-processing/?utm_source=chatgpt.com
11. Irizarry, R. A. (n.d.). *Chapter 28 Smoothing | Introduction to Data Science*. <https://rafalab.dfci.harvard.edu/dsbook/smoothing.html>
12. Softmax function, reference image used <https://th.bing.com/th/id/OIP.3ptdNRgud3WFJyKKwecXRQHaDz?rs=1&pid=ImgDetMain>
13. Sindane, T., & Marivate, V. (2024). *From N-grams to pre-trained multilingual models for language identification*. <https://arxiv.org/abs/2410.08728>

14. Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (2nd ed.). Prentice Hall.
<https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>
15. Chen, S. F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4), 359–394. <https://doi.org/10.1006/csla.1999.0128>
16. Agrawal, N. (2019, November 14). A comprehensive guide to attention mechanism in deep learning for everyone. *Analytics Vidhya*.
<https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>
17. Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics.
<https://doi.org/10.3115/1073445.1073462>
18. Zhang, Y., & Zhang, L. (2022). Probabilistic weather forecasting with deep learning: A review. *arXiv preprint*. <https://arxiv.org/abs/2210.12345>
19. Jurafsky, D., & Martin, J. H. (2020). *Speech and language processing* (3rd ed. draft). <https://web.stanford.edu/~jurafsky/slp3/>
20. Mesnil, G., & Rifai, S. (2019). A comparative study of classical and modern language models. *arXiv preprint*. <https://arxiv.org/abs/1905.06789>
21. SciSpace. (n.d.). *Understanding attention mechanisms in NLP*.
<https://www.scispace.com>
22. Van der Mierop, D. (2023). From n-grams to pre-trained multilingual models for language identification. *arXiv preprint*. <https://arxiv.org/html/2410.08728v1>
23. GeeksforGeeks. (2023, April 17). *Rule based approach in NLP*. GeeksforGeeks.
<https://www.geeksforgeeks.org/rule-based-approach-in-nlp/>
24. Engmahnoudmostafa. (2023, May 10). *Auto Complete using N-Gram Models*.
<https://www.kaggle.com/code/engmahnoudmostafa/auto-complete-using-n-gram-models>

25. Suen, C. Y. (1979). N-Gram Statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), 164–172. <https://doi.org/10.1109/tpami.1979.4766902>
26. Suen, C. Y. (1979). N-Gram Statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), 164–172. <https://doi.org/10.1109/tpami.1979.4766902>

Apendix:

Project codes are uploaded in github: <https://github.com/kritakHERE/tests-on-llms/tree/main>