Dauphine | PSL
UNIVERSITÉ PARIS

Evaluating Autoregressive Decoding Strategies for Optimal Quality–Diversity Tradeoff in Text Generation

Julien Bernardi

Supervisor : Prof. Pierre Wolinski

This report documents the methodology, implementation, and evaluation of several autoregressive decoding techniques applied to LSTM-based text generation models.
We aim to identify the best compromise between output quality and diversity.

May 25, 2025

Project Structure

The repository is organized as follows :

models/	Contains saved model checkpoints (`.pt` files) for our LSTM baseline.
utils/	Stores all generated figures (JPEG/PNG) used in the report : bar charts, scatter plots, heatmaps, beam-search trees, etc.
wikitext2/	Holds the raw WikiText-2 dataset split into `train.txt`, `valid.txt` and `test.txt`.
data_processing.py	Implements corpus cleaning, `<unk>`-filtering, tokenization, vocabulary building and tensor creation.
decoding_strategies.py	Defines the four decoding algorithms : greedy, beam search, top-k sampling and nucleus (top-p) sampling.
evaluation.py	Provides functions to compute automatic metrics (BLEU, ROUGE, Levenshtein distance, perplexity).
generate_evaluations_plots.py	Automates the creation of comparative performance plots for each decoding strategy.
generate_text.py	Generates sample continuations from the trained model using any specified decoding strategy.
generate_visualizations.py	Produces illustrative plots (probability distributions, dynamic cutoffs, cumulative curves) for each method.
model.py	Defines the LSTM architecture : embedding layer, stacked LSTM cells, dropout and output projection.
notebook_visualization.ipynb	Interactive Jupyter notebook for rapid prototyping and previewing outputs before formalizing scripts.
plot_utils.py	Utility functions for plotting (bar charts, scatter plots, heatmaps) reused across scripts.
train.py	Main training script : data loading, model training loop, checkpoint saving and logging.

Contents

Introduction

Automatic text generation lies at the heart of many modern NLP applications—from conversational agents and machine translation to content summarization and creative writing. As models have become more powerful, the challenge has shifted from crafting ever-larger architectures to how we extract coherent, informative, and diverse text from them. Different decoding algorithms (greedy, beam search, top-k, nucleus sampling) offer distinct trade-offs between fidelity to the model's most probable output and the lexical variety that makes generated language engaging and human-like.

In this project, we implement a simple LSTM-based language model and systematically evaluate these decoding strategies on a standard benchmark. Our goal is to identify which algorithms best balance quality (fluency and relevance) and diversity (avoiding repetition and promoting creativity).

1 Data Collection and Pre-processing

1.1 Dataset Selection and Acquisition

We chose the **WikiText-2** corpus because it is a well-known benchmark for autoregressive language modeling : it contains roughly 2 million tokens drawn from "featured" Wikipedia articles, ensuring high linguistic quality and a variety of topics. Its moderate size makes it feasible to train LSTM models in a reasonable time on consumer-grade hardware, while still exhibiting sufficient vocabulary richness to stress our decoding strategies.

1.2 Data Cleaning and Filtering

When loading the raw WikiText-2 corpus, we observed a total of 54 625 occurrences of the special token `<unk>`, indicating out-of-vocabulary words. To prevent this noise from degrading model training and text generation, we applied a simple *line-level filter* : any line where more than 3 % of tokens were `<unk>` was discarded.

— Total lines processed : 23 767

— Lines dropped : 7 055 (29.7 % of all lines)

— Lines kept : 16 712 (70.3 % of all lines)

— Total `<unk>` tokens encountered : 54 625

— `<unk>` tokens removed (in dropped lines) : 36 440

1.3 Tokenization and final pre-processing steps

In WikiText-2 each line represents either a short paragraph or an article heading. To preserve these natural document boundaries, we wrap each line with a begin-of-sequence and end-of-sequence marker. During the first pass over `train.txt`, we :

— Prepend `<BOS>` and append `<EOS>` to each line to mark paragraph or title boundaries.

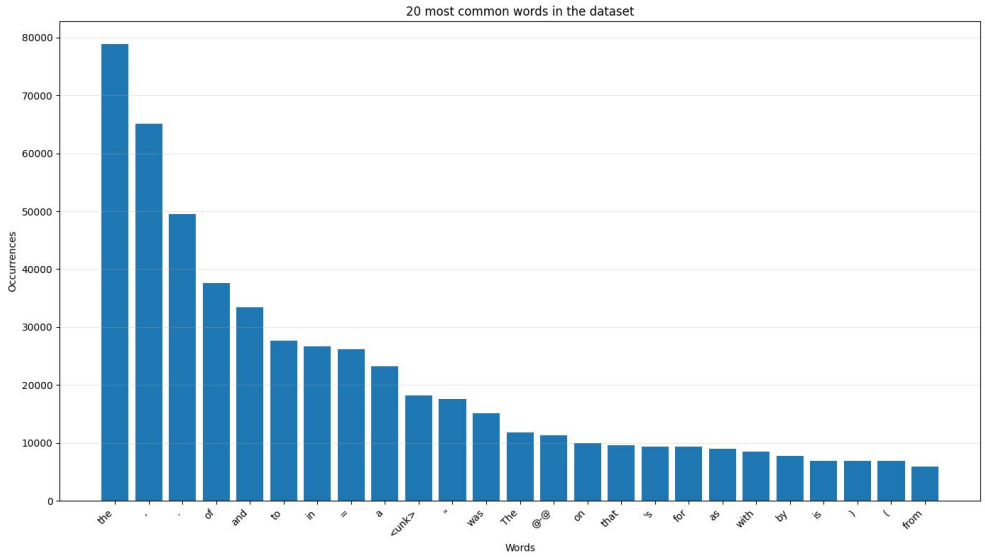
— Count occurrences of every token using our `Dictionary` class, yielding a vocabulary of 31 693 entries (including `<PAD>`, `<UNK>`, `<BOS>`, `<EOS>`).

On the second pass, we convert each marked line into a sequence of token IDs :

— Map each in-vocabulary word to its index ; out-of-vocabulary words default to `<UNK>`.

— Concatenate all line-level sequences (now including `<BOS>`...`<EOS>`) into a single tensor.

This approach preserves the contextual integrity of paragraphs and headings, retains full lexical richness without arbitrary truncation, and handles unknown words gracefully. For training, we then slice this long token stream into sliding windows of fixed length (e.g. 35 tokens) with a step size of 1, pairing each window with its next-token target. Finally, we organize these sequences into mini-batches (batch size = 32) for efficient autoregressive LSTM training.



Top 20 most common words in the cleaned WikiText-2 corpus.

2 Model Architecture and Training

2.1 Embedding Layer Design

We begin by mapping each token index into a dense vector of size 300 via a learned `nn.Embedding` layer. This transforms sparse integer IDs into continuous representations that capture distributional semantics. During training, these embeddings are updated jointly with the rest of the network.

2.2 Basic LSTM Setup

Our core language model is a multi-layer LSTM (`nn.LSTM`) with the following configuration :

— **Input dimension :** 300 (embedding size)

— **Hidden dimension :** 512 per layer

— **Number of layers :** 2 stacked LSTM cells

— **Dropout :** 0.5 between LSTM layers (disabled if only one layer)

At each time step, the LSTM consumes an embedding and updates its hidden and cell states. We apply a dropout layer on the LSTM outputs to regularize the network before the final projection.

2.3 Output Projection and Loss

A fully connected layer (`nn.Linear`) maps the LSTM's hidden vectors (size 512) back to the vocabulary dimension. The model therefore outputs a tensor of shape `[batch, seq_len, vocab_size]`. During training, we reshape this to `[batch*seq_len, vocab_size]` and compute the standard cross-entropy loss against the true next-token indices.

2.4 Training Loop and Hyperparameters

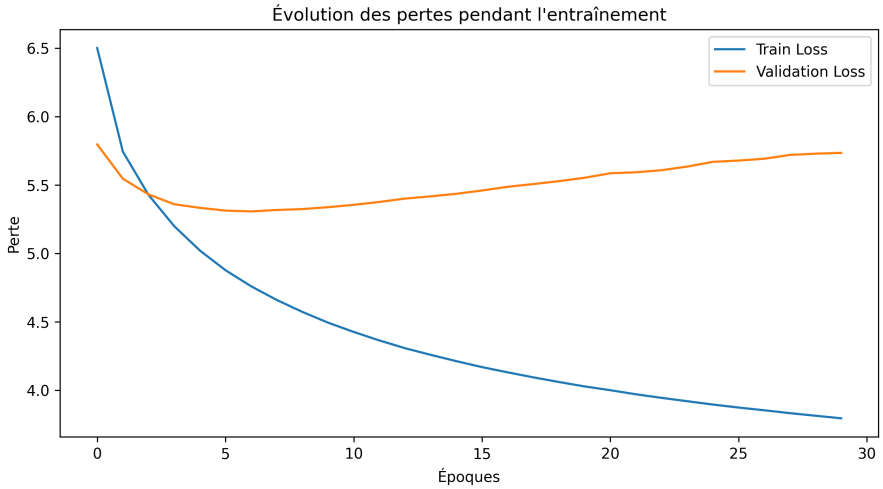
We train using the Adam optimizer (learning rate 1e-3) and gradient clipping (max norm = 5) to prevent exploding gradients. Our protocol :

— **Batch size :** 32 sequences

— **Sequence length :** 35 tokens

— **Epochs :** 30

The loss curves show that while the training loss continues to decrease steadily, the validation loss reaches its minimum around epoch 7 before slowly rising : an indication of overfitting. Although our model is clearly overfitting, we simply select and retain the model checkpoint from the best-performing epoch. We could of course, apply techniques to reduce overfitting, but this is beyond the scope of our current project, which focuses on evaluating decoding strategies using a fixed LSTM baseline.



Training and validation loss over 30 epochs.

This setup is thus sufficient to compare and evaluate different autoregressive decoding strategies under consistent model conditions.

3 Autoregressive Decoding Strategies

3.1 Greedy Decoding

Greedy decoding selects at each time step the single token with highest predicted probability :

$$w_t = \arg\max_w \; p(w \mid w_{<t}) \, .$$

This yields a fast, deterministic generation but often sacrifices diversity : the model follows its most confident path and can get stuck in repetitive loops.

Hypothesis. Greedy decoding will produce high local coherence (quality) in short sequences but suffer from low lexical variety and may generate bland or repetitive text over longer spans.

3.2 Beam Search

Beam search maintains a set (beam) of the top K partial hypotheses at each step, extending each by all possible next tokens and retaining the best K sequences according to cumulative log-probability (optionally length-normalized).

$$\text{score}(\text{seq}) = \sum_t \log p(w_t \mid w_{<t}) \, .$$

By exploring multiple paths, beam search balances local and global quality.

Hypothesis. With moderate beam widths (e.g. $K = 5$), we expect improved overall coherence and fewer dead-end paths compared to greedy, but still limited diversity since only high-probability sequences are considered.

3.3 Top-k Sampling

Top-k sampling truncates the vocabulary to the k most probable tokens at each step, renormalizes their probabilities, and samples one token at random :

$$\tilde{p}(w) = \frac{p(w)}{\sum_{i \in \text{top-}k} p(i)}, \quad w \sim \tilde{p}(\cdot).$$

This injects controlled randomness.

Hypothesis. As k increases, diversity should rise (more varied word choices), but quality may degrade if unlikely tokens enter the candidate set ; small k (e.g. 10–50) should strike a balance.

3.4 Nucleus (Top-p) Sampling

Nucleus sampling selects the smallest set of tokens whose cumulative probability exceeds threshold p (e.g. 0.9), then samples from this dynamic subset.

$$\mathcal{V}_p = \{w : \sum_{i \leq w} p(i) \geq p\}, \quad w \sim \frac{p(w)}{\sum_{i \in \mathcal{V}_p} p(i)}.$$

This adapts the candidate pool size at each step.

Hypothesis. Nucleus sampling will outperform fixed top k at maintaining coherence when distributions are flat, while still offering high diversity—ideal for generating creative yet plausible continuations.

3.5 Quick visual analysis

Comparison of decoding strategies for the prompt “The President of the United States”

Strategy	**Generated Text**
Greedy	The President of the United States entered the United States, and the United States entered the United States.
Top-k (k=10)	The President of the United States Navy, in particular the British Party in the late 1930s. The `<unk>` is also known as the Ancient Territory. He was also the most important known of the NDH regime, which is also a large state of the NDH and western American Republic.
Beam Search (width=3)	The President of the United States entered World War II. It was the first of the United States Navy in the United States.
Nucleus (p=0.9)	The president of the United States entered the evening on August 5 ; the 21st Brigade had gained off to present the Chinese strength and dissipating an average hurricane of about 20 000 lb.

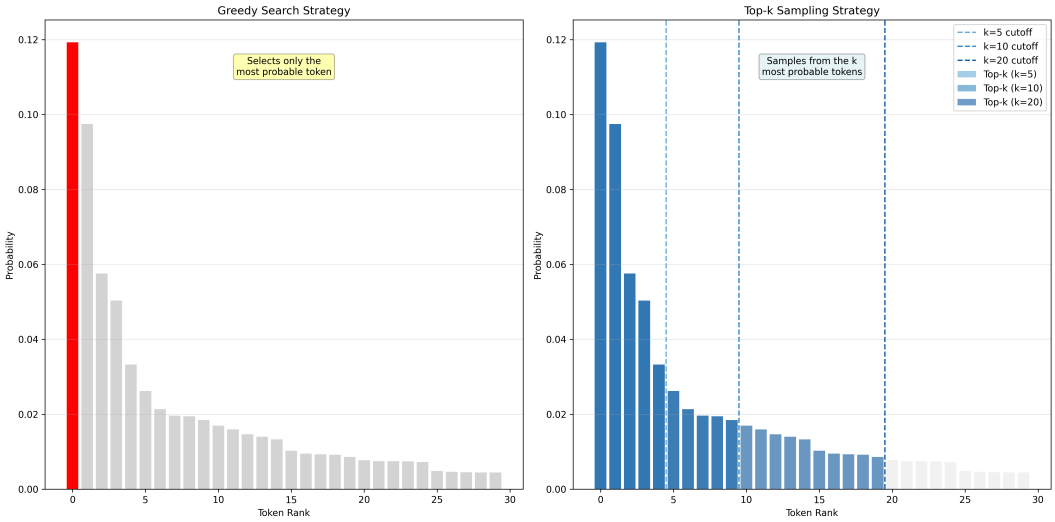
— **Greedy decoding** always picks the highest-probability token, resulting in very repetitive and unsurprising text.

— **Top-k sampling** (k=10) introduces greater lexical variety.

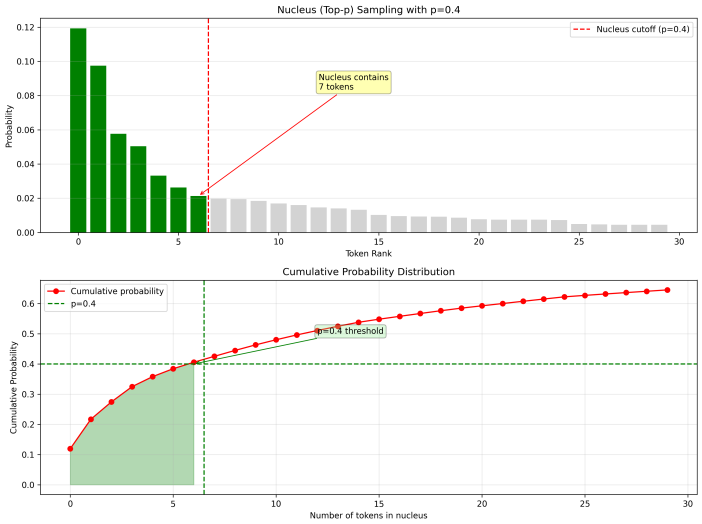
— **Beam Search** (width=4) improves overall coherence by exploring multiple hypotheses in parallel, while remaining relatively conservative.

— **Nucleus sampling** (p=0.9) offers the highest creativity and diversity, at the cost of occasional factual errors or atypical constructions.

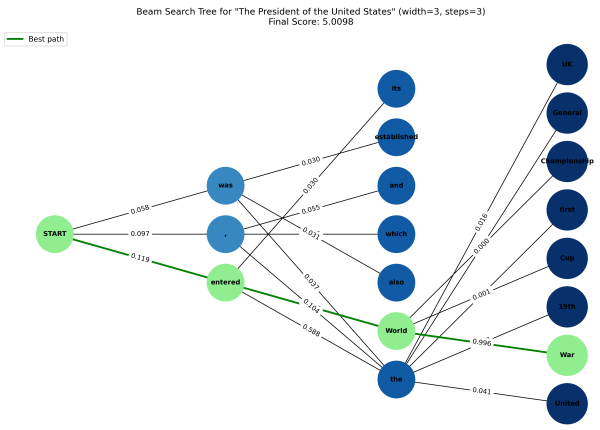
These examples illustrate the quality–diversity trade-off.We will analyze these trade-offs quantitatively using established evaluation metrics in the next section.



(a) Greedy Decoding vs Top-k Sampling



(b) Nucleus (Top-p) Sampling with $p = 0.4$



(c) Beam Search Tree (width=3, steps=3)

Visualization of Decoding Strategies. (a) Comparison of Greedy decoding and Top-k sampling on the same distribution. (b) Illustration of nucleus sampling showing the dynamic cutoff and the cumulative probability distribution.(c) Structured beam search tree illustrating how hypotheses are expanded and pruned.

4 Evaluation Metrics and Analysis

4.1 BLEU

BLEU (Bilingual Evaluation Understudy) is a widely used *precision*-oriented metric for automatically evaluating the quality of generated text against one or more human reference texts. Rather than comparing full sentences, BLEU operates on the level of n-grams (contiguous sequences of n tokens), rewarding the generator for producing the same n-grams as the reference.

Step 1 : Extract and count n-grams. For each n from 1 to N (commonly $N = 4$), list all n-grams in the hypothesis $\mathcal{H}$ and in each reference $\mathcal{R}_i$. Denote by $\text{count}_{\mathcal{H}}(g)$ the number of occurrences of n-gram g in the hypothesis, and by $\text{count}_{\mathcal{R}_i}(g)$ its count in reference i.

Step 2 : Clipped match counts. To avoid overcounting repeated n-grams, BLEU uses *clipped counts*. For each g,

$$\text{match}_n = \sum_g \min\Big(\text{count}_{\mathcal{H}}(g),\ \max_i \text{count}_{\mathcal{R}_i}(g)\Big).$$

Here, we compare the hypothesis's count only up to the maximum number of times g appears in any single reference.

Step 3 : Precision per order. The n-gram precision p_n is then

$$p_n = \frac{\text{match}_n}{\sum_g \text{count}_{\mathcal{H}}(g)},$$

i.e. the fraction of the hypothesis's n-grams that appear in the references.

Step 4 : Brevity penalty. BLEU applies a penalty to discourage very short hypotheses that achieve high precision by omitting content. Let

$$c = |\mathcal{H}| \quad \text{(length of hypothesis)}, \qquad r = \text{argmin}_{r_i} \big| |\mathcal{R}_i| - c \big| \quad \text{(reference length closest to } c\text{)}.$$

Define

$$\text{BP} = \begin{cases} 1, & c > r, \\ \exp\big(1 - \frac{r}{c}\big), & c \le r. \end{cases}$$

Thus, if the hypothesis is shorter than the reference, BP$<$1 reduces the overall score.

Step 5 : Final BLEU score. Combining all orders equally (uniform weights $1/N$), the BLEU score is

$$\text{BLEU} = \text{BP} \ \times \ \exp\Big(\tfrac{1}{N} \sum_{n=1}^{N} \ln p_n\Big).$$

4.2 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of recall-based metrics designed to measure how much of the content in a human reference summary (or text) is "covered" by a candidate (generated) text. Unlike BLEU, which emphasizes precision, ROUGE focuses on recall of units such as n-grams or subsequences. Two of the most widely used variants are ROUGE-N and ROUGE-L.

ROUGE-N (n-gram recall) ROUGE-N computes the fraction of n-grams in the reference $\mathcal{R}$ that also appear in the hypothesis $\mathcal{H}$:

$$\text{ROUGE}-N = \frac{\sum_{g \in \mathcal{R}} \min\big(\text{count}_{\mathcal{R}}(g),\, \text{count}_{\mathcal{H}}(g)\big)}{\sum_{g \in \mathcal{R}} \text{count}_{\mathcal{R}}(g)}.$$

— *Numerator :* total number of reference n-grams that the hypothesis successfully reproduces (clipped to avoid over-counting).

— *Denominator :* total number of n-grams in the reference.

— *Interpretation :* a value of 1 means the hypothesis covers every n-gram in the reference ; lower values indicate missing content.

ROUGE-L (longest common subsequence) ROUGE-L is based on the length of the Longest Common Subsequence (LCS) between hypothesis and reference, capturing in-order matches without requiring contiguity :

$$\text{LCS}(\mathcal{H}, \mathcal{R}) = \max\{\, |S| : S \text{ is a subsequence of both } \mathcal{H}, \mathcal{R}\}.$$

Define

$$R_L = \frac{\text{LCS}(\mathcal{H}, \mathcal{R})}{|\mathcal{R}|}, \quad P_L = \frac{\text{LCS}(\mathcal{H}, \mathcal{R})}{|\mathcal{H}|}.$$

ROUGE-L is then the F_β measure of (R_L, P_L), typically with $\beta = 1$:

$$\text{ROUGE}-L = \frac{(1+\beta^2)\, R_L\, P_L}{R_L + \beta^2\, P_L} \quad (\beta = 1).$$

— R_L (recall) : proportion of the reference sequence "covered" by a longest subsequence.

— P_L (precision) : proportion of the hypothesis sequence explained by that same subsequence.

— The F-measure balances R_L and P_L, rewarding both coverage and conciseness.

Key points

— ROUGE-N focuses purely on *coverage* of reference n-grams (recall), making it sensitive to missing content.

— ROUGE-L accounts for *sequence fluency* by rewarding longer in-order matches (even if not contiguous).

4.3 Comparative Results

Reference : "The development of artificial intelligence has transformed various industries including healthcare, and transportation."

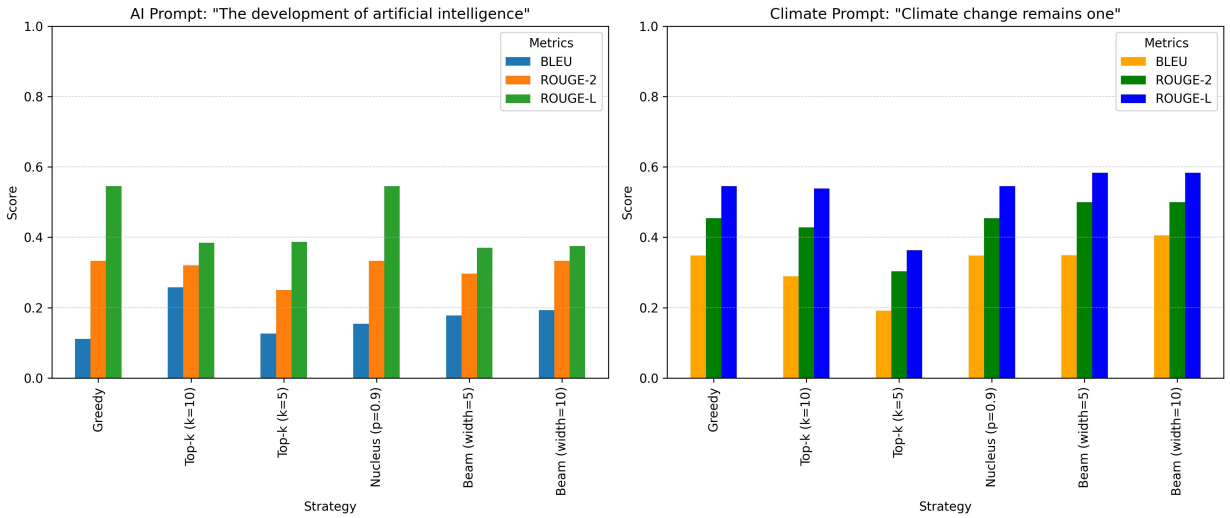
Decoding Strategies on Prompt "The development of artificial intelligence"

Strategy	Excerpt of Generated Text	BLEU	ROUGE-2 F1	ROUGE-L F1
Greedy	The development of artificial intelligence, the <unk> of the <unk>, ...	0.1109	0.3333	0.5455
Top-k (k=10)	The development of artificial intelligence had been made to use a number of players.	0.2575	0.3200	0.3846
Top-k (k=5)	The development of artificial intelligence is a common @-@ based version of the game's largest, <unk>, ...	0.1263	0.2500	0.3871
Nucleus (p=0.6)	The development of artificial intelligence, the <unk> of the <unk>, ...	0.1545	0.3333	0.5455
Beam (width=5)	The development of artificial intelligence was a <unk> of the United States Coast Guard ...	0.1779	0.2963	0.3704
Beam (width=10)	The development of artificial intelligence has been described as a "<unk>". It was also released ...	0.1929	0.3333	0.3750

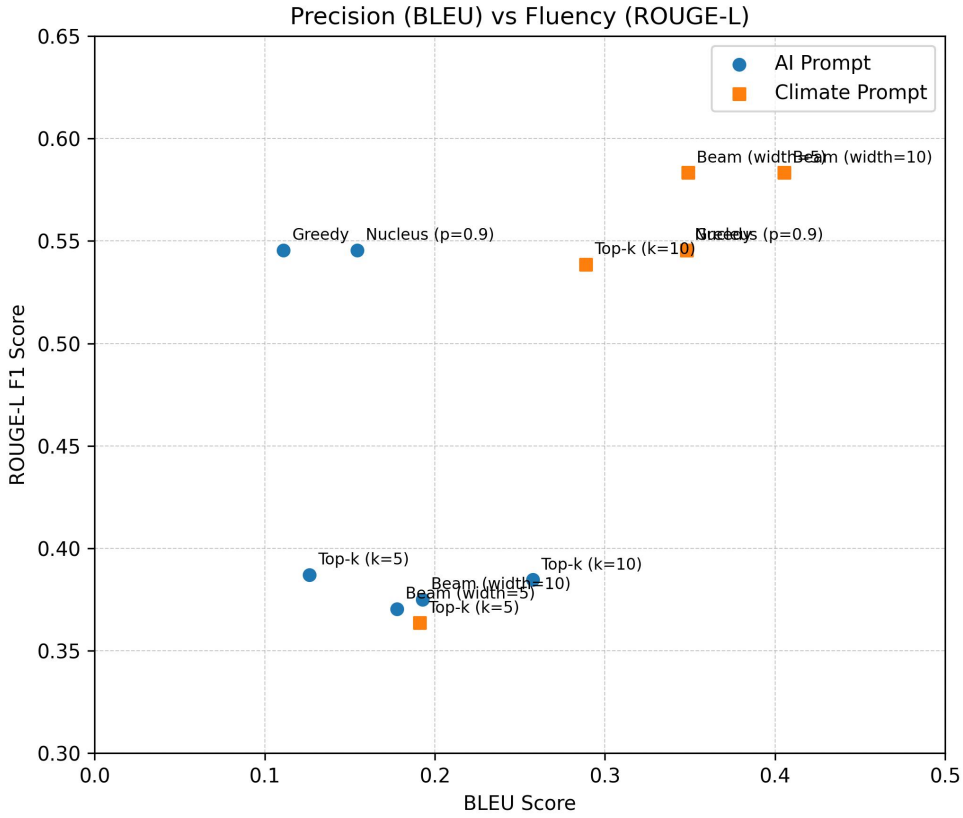
Reference : "Climate change remains one of the biggest challenges facing humanity in history."

Decoding Strategies on Prompt "Climate change remains one"

Strategy	Excerpt of Generated Text	BLEU	ROUGE-2 F1	ROUGE-L F1
Greedy	Climate change remains one of the most important <unk> of the world.	0.3479	0.4545	0.5455
Top-k (k=10)	Climate change remains one of the best, or one of its original, in the late 1980s.	0.2889	0.4286	0.5385
Top-k (k=5)	Climate change remains one of the most important and the best @-@ time game. The song is not released on a tour.	0.1912	0.3030	0.3636
Nucleus (p=0.6)	Climate change remains one of the most important <unk> of the world.	0.3479	0.4545	0.5455
Beam (width=5)	Climate change remains one of the most common <unk> in the United States.	0.3490	0.5000	0.5833
Beam (width=10)	Climate change remains one of the most common events in the United States.	0.4053	0.5000	0.5833



Comparison of BLEU, ROUGE-2 and ROUGE-L for each decoding strategy on two prompts.



Precision (BLEU) vs. fluency (ROUGE-L) for each decoding strategy, shown for both the AI prompt (circles) and the climate prompt (squares).

Global Interpretation :

— **Beam search** consistently achieves the highest BLEU and ROUGE-2 scores on both prompts when the beam width is increased to 10, confirming its ability to capture high-probability sequences and improve n-gram overlap.

— **Greedy** and **Nucleus (p=0.6)** yield identical ROUGE-L in both cases (0.5455), indicating that they both optimize local fluency but risk repetition or limited diversity.

— **Top-k sampling** shows a trade-off : with $k = 10$, BLEU is relatively high on the AI prompt (0.2575) but drops on the climate prompt (0.2889), while ROUGE-L remains moderate ; reducing k to 5 further increases diversity but penalizes precision and recall metrics.

— Overall, *beam search with width 10* offers the best balance of precision and coverage, while *top-k* and *nucleus* sampling provide more varied but less predictable outputs.

Conclusion

In this work, we implemented a simple LSTM-based language model and systematically compared four popular autoregressive decoding strategies—greedy decoding, beam search, top-k sampling, and nucleus (top-p) sampling—on two distinct prompts. Our quantitative evaluation (BLEU, ROUGE-2, ROUGE-L) and visual analyses revealed the following key findings :

— **Beam search** with a sufficiently large beam (width=10) consistently delivers the highest precision and recall (BLEU, ROUGE-2), at the cost of somewhat reduced diversity.

— **Greedy decoding** and **nucleus sampling** (p=0.6) produce comparably high ROUGE-L scores, indicating strong local fluency, but tend to repeat common tokens and lack variety.

— **Top-k sampling** allows flexible control over diversity via k : larger k improves BLEU at the expense of coherence, while smaller k enhances novelty but lowers overlap with references.

These results illustrate the fundamental trade-off between *quality* (faithfulness to high-probability sequences) and *diversity* (lexical and structural variation). By selecting beam search for applications requiring maximal accuracy and top-k/nucleus sampling for more creative generation, practitioners can tailor decoding to specific use cases.