

1.5 Comparative Analysis [1 mark]

- **Compare your two trained models (Category I vs Category II):**
 - Dataset size, vocabulary, context predictability
 - Model performance (loss curves, qualitative generations)
 - Embedding visualizations
- **Summarize insights on how natural vs structured language differs in learnability.**

1. For category 1:

Dataset: [Leo Tolstoy's War and Peace](#)

For category 2:

Dataset: Python code from repository:

<https://github.com/wangshusen/GIANT-Python-Code.git>

Vocabulary:

For category 1:

Total number of lines (sentences): 30660

Total number of words (with duplicates): 563946

Total number of unique words (vocabulary size): 19764

For category 2:

Total tokens: 13628

Unique tokens: 521

Top 10 frequent and Bottom 10 frequent(least frequent) words/tokens are listed in jupyter notebook file task1.ipynb and task1_python.ipynb

Model Performance:

Loss:

For category 1 dataset: Epoch ran: 100

At the end of the 100th epoch, it converged for around 0.4.

For category 2 dataset, Epoch ran: 1000

At the end of the 1000th epoch, it converged around 1.8456.

The text-based model converged faster with a lower loss (~0.4) compared to the code-based model (~1.85), mainly due to fundamental differences in data structure and predictability. Natural language is redundant and context-tolerant, allowing the model to generalize patterns easily and achieve smoother convergence. In contrast, programming languages like Python are highly structured, precise, and less repetitive — a single incorrect token can invalidate syntax. This strict dependency and lower redundancy make code prediction more complex, leading to slower convergence and higher final loss. Overall, the results highlight that natural language is easier

for models to learn statistically, while structured code demands greater syntactic precision and yields higher training loss.

Embedding Visualisations Difference:

Feature	Code Embeddings	Text Embeddings
Distribution	Sparse; clearly defined token groups per programming category.	Dense, circular, continuous — natural word embeddings form smooth manifolds.
Center density	Many tokens cluster near the center (common syntax), while rare symbols are peripheral.	High density in the center — common words dominate, specialized words lie on the edges.
Axes spread	Moderate (≈ -6 to $+6$), structured per token type.	Broader coverage (≈ -7 to $+7$), indicating richer contextual relationships.

Code embeddings are **rule-bound, discrete, and category-separated** (syntax-driven).

Text embeddings are **semantically rich, continuous, and diffuse** (meaning-driven).

Learneability:

Natural language is **easier to learn statistically** because it is rich in patterns, redundancy, and contextual flexibility. Models can infer meaning even from partial information, as small prediction errors rarely distort overall understanding. Its structure is probabilistic — many valid word combinations exist — allowing smooth generalization across similar contexts.

Structured languages like Python, however, are **harder to learn** because they follow **strict syntax and logical rules**. Each token has a specific role and limited valid successors; a single incorrect prediction breaks grammatical validity. There is little redundancy, so models must learn exact structural dependencies rather than semantic associations