Отчет студента 325-й группы Зверева Дмитрия о выполненном задании “Предсказание следующего элемента”, п.2 (предсказание символов)

Критерии оценки качества программы

Метрики качества:

- **Simple Accuracy** — доля совпадающих символов;
- **3-Gram Accuracy** — доля совпадающих 3-грамм.

Когерентность текста: проверка, не порождает ли модель слишком однотипные фразы или последовательности и анализ, насколько осмысленно выглядит сгенерированный текст.

Также определим, на что мы будем смотреть при обучении модели:

- **графики функции потерь и точности;**
- **скорость сходимости;**
- **генерация текста на промежуточных этапах обучения**, т.е. проверка, как сгенерированный текст улучшался с ростом числа эпох.

Варианты экспериментов

Будем экспериментировать со следующими составами текста *cat.txt*:

- *буквы с пробелами*;
- *буквы, а также пробелы и дефисы*;
- *буквы, пробелы, дефисы и точки*;
- *полный набор*: буквы, пробелы, дефисы, точки, запятые.

Модификация предоставленной на лекции программы

Добавим возможность сохранять разные наборы символов (пробелы, дефисы, точки, запятые) в зависимости от эксперимента. Это управляется через словарь *variants*:

```
variants = {
    "letters_only": "А-яё ",
    "letters_spaces_dashes": "А-яё\- ",
    "letters_spaces_dashes_dots": "А-яё\-\. ",
    "full_set": "А-яё\-\. ,"
}
```

Дадим возможность переменной *num_characters* определять размер автоматически на основе уникальных символов в тексте:

```
num_characters = len(set(text))
```

Также добавим *метрики оценки*: Simple Accuracy (доля совпадающих символов между предсказанным и эталонным текстом) и 3-Gram Accuracy (доля совпадающих триграмм).

Помимо прочего, сделаем код структурированным (с помощью написания функций) и работающим для всех указанных выше случаев одновременно за один запуск.

Результаты

После запуска программы на cat.txt получаем следующие результаты:

```
=== Variant: letters_only ===
Generated: кот на то было не подной подной подной подной подной
Simple Accuracy: 0.0566
3-Gram Accuracy: 0.1373

=== Variant: letters_spaces_dashes ===
Generated: кот на на на на на на на на на на на на на на на н
Simple Accuracy: 0.0566
3-Gram Accuracy: 0.0392

=== Variant: letters_spaces_dashes_dots ===
Generated: котёнка и прина столько дел котёнка и прина столько д
Simple Accuracy: 0.0755
3-Gram Accuracy: 0.0980

=== Variant: full_set ===
Generated: кот вершенный принние серего от вершенный принние сер
Simple Accuracy: 0.0189
3-Gram Accuracy: 0.1373

=== Summary of Results ===
letters_only: Simple Accuracy = 0.0566, 3-Gram Accuracy = 0.1373
letters_spaces_dashes: Simple Accuracy = 0.0566, 3-Gram Accuracy = 0.0392
letters_spaces_dashes_dots: Simple Accuracy = 0.0755, 3-Gram Accuracy = 0.0980
full_set: Simple Accuracy = 0.0189, 3-Gram Accuracy = 0.1373
```

Проанализируем каждый из случаев:

- *буквы с пробелами*
    - Generated: "кот на то было не подной подной подной подной подной"
    - Simple Accuracy: 0.0566 (5.66%)
    - 3-Gram Accuracy: 0.1373 (13.73%)

Модель работает только с буквами, что упрощает задачу, но теряет важные границы предложений (точки, дефисы). Это приводит к генерированию монотонных фраз, таких как "подной подной". Simple Accuracy низкая, так как предсказания моделей часто расходятся с реальными символами. 3-Gram Accuracy выше, поскольку модель может генерировать короткие фрагменты, которые частично совпадают с оригинальным текстом, но этого недостаточно для генерации осмысленных длинных последовательностей.

- *буквы, а также пробелы и дефисы*
    - Generated: "кот на на на на на на на на на на на на на на на на"
    - Simple Accuracy: 0.0566 (5.66%)
    - 3-Gram Accuracy: 0.0392 (3.92%)

Добавлены дефисы, но модель, похоже, не использует их, генерируя очень повторяющиеся последовательности ("на на на"). Низкая Simple Accuracy показывает, что модель не справляется с предсказанием отдельных символов. 3-Gram Accuracy еще ниже, чем у варианта выше, что может быть связано с переобучением на однообразных данных. Модель просто повторяет предыдущие слова без учета контекста.

- *буквы, пробелы, дефисы и точки*
    - Generated: "котёнка и прина столько дел котёнка и прина столько"
    - Simple Accuracy: 0.0755 (7.55%)
    - 3-Gram Accuracy: 0.0980 (9.80%)

Сохранены точки, которые помогают модели выделять границы предложений. В отличие от предыдущих вариантов, модель генерирует немного более осмысленный текст, но повторения "прина столько" указывают на ограничение в способности обрабатывать контекст. Simple

Accuracy немного выше, но все еще низкая, что говорит о том, что предсказание отдельных символов остается сложным. 3-Gram Accuracy также улучшилась, но по-прежнему достаточно низка.

- *полный набор*
    - Generated: "кот вершенный принние серего от вершенный принние сер"
    - Simple Accuracy: 0.0189 (1.89%)
    - 3-Gram Accuracy: 0.1373 (13.73%)

Полный набор символов, включая запятые и точки. Модель генерирует фразы с неплохим разнообразием, но из-за сложности данных (пунктуация и разнообразие символов) точность сильно падает. Очень низкая Simple Accuracy указывает на проблемы с предсказанием отдельных символов. 3-Gram Accuracy на уровне первоначального варианта, что свидетельствует о частичном совпадении повторяющихся фрагментов.

Общие выводы

1. Наиболее стабильный результат по 3-граммам у варианта с *буквами и пробелами* и *полным набором* (13.73%), но это происходит за счет генерации однообразных фрагментов;
2. Вариант *буквы, пробелы, дефисы и точки* показывает сбалансированный результат, т.к. использование точек помогает модели понимать границы предложений;
3. Регулярные повторения в сгенерированных текста (например, "подной подной", "на на на") указывают на недостаток контекста для модели;
4. Сложность *полного набора* символов ухудшает точность предсказаний из-за переобучения на шумных данных.