ДОКУМЕНТАЦИЯ, СОДЕРЖАЩАЯ ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ХАРАКТЕРИСТИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

PROMT Neural Translation Server – Training Addon (для ОС Linux)

И ИНФОРМАЦИЮ, НЕОБХОДИМУЮ ДЛЯ УСТАНОВКИ И ЭКСПЛУАТАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

PROMT Neural Translation Server (Linux) – Training Addon 23

Руководство администратора

PROMT Neural Translation Server (Linux) – Training Addon 23

Руководство администратора

Россия, 199155,

Санкт-Петербург, Уральская ул., д. 17, лит. Е, кор. 3.

E-mail: common@promt.ru

support@promt.ru

Internet: https://www.promt.ru

https://www.translate.ru

Телефон/ факс: +7 812 655-0350

Оглавление

1. О документации

1.1. Состав документации

Документация *PROMT Neural Translation Server (Linux) – Training Addon* содержит сведения об установке указанного компонента, а также информацию, необходимую для настройки (обучения) моделей нейронного машинного перевода (NMT-моделей), использующихся в переводчике PROMT, с помощью корпуса параллельных двуязычных текстов.

Данное руководство предназначено для пользователей переводчика *PROMT Neural Translation Server* (*PNTS*), включая его модификации, осуществляющих настройку NMT-моделей посредством дополнительной функциональности переводчика, реализованной в компоненте *PROMT Neural Translation Server (Linux) – Training Addon*.

PROMT Neural Translation Server — это единый набор программных модулей, позволяющих получить нейронный машинный перевод через веб-интерфейс.

Подробные сведения о работе с переводчиком *PNTS* см. в руководстве пользователя или в руководстве администратора PNTS.

1.2. Условные обозначения

В документации используются следующие условные обозначения:

— советы и рекомендации;

— важные замечания;

— примечания или дополнительная информация;

— ссылка на подробные сведения в руководстве.

2. Введение

В PROMT Neural Translation Server используется нейронный машинный перевод (Neural Machine Translation — NMT) — разновидность машинного перевода, в основе которого лежит механизм двунаправленных рекуррентных нейронных сетей (Bidirectional Recurrent Neural Networks), построенный на матричных вычислениях. Нейронный перевод требует для обучения параллельные корпуса, позволяющие сравнить автоматический перевод с эталонным «человеческим», при этом в процессе обучения он оперирует не отдельными фразами и словосочетаниями, а целыми предложениями.

2.1. Назначение *PROMT Neural Translation Server (Linux) – Training Addon*

PROMT Neural Translation Server (Linux) – Training Addon (далее – *PNTS Linux Training Addon)* является дополнением для программного продукта *PROMT Neural Translation Server* (далее *PNTS*), включая его модификации, и предназначен для изменения рабочих характеристик нейронного машинного перевода в программном продукте за счет автоматизированной настройки (обучения, тюнинга) моделей нейронного машинного перевода (NMT-моделей). Автоматизированная настройка производится с помощью корпуса параллельных двуязычных текстов, предоставляемых пользователем.

Компонент *PNTS Linux Training Addon* расширяет функциональность установленного у пользователя программного продукта *PNTS* за счет дополнительной функциональной возможности автоматической настройки моделей нейронного машинного перевода.

PNTS Linux Training Addon представляет собой набор утилит командной строки, которые последовательно запускаются с помощью bash-скриптов.

Входными данными для системы настройки являются NMT-модель (в виде zip-архива) и корпус параллельных текстов.

Результатом работы является настроенная по текстовому корпусу новая модель (в виде zip-архива), которую необходимо загрузить в PNTS.

2.2. Помощь и сопровождение

В случае возникновения проблем при работе с программным продуктом следует обратиться в отдел технической поддержки по телефону или отправить сообщение по электронной почте. При этом укажите следующее:

- языковые пары, которые используются с данным программным продуктом;
- основные характеристики компьютера: тип процессора, объем оперативной памяти, объем свободного места на жестком диске, наличие сети, GPU, версия драйверов видеокарты/ видеокарт, версия CUDA;
- характеристики используемого ПО: имя и версия дистрибутива Linux (например, Ubuntu 20.04.1 LTS), имя и версия ядра Linux (результат вызова команды uname -a);

- суть проблемы и действия, предшествовавшие ее появлению;
- действия, предпринятые для решения данной проблемы;
- при получении сообщения об ошибке — его точный текст или снимок экрана с этим сообщением.

При обращении в отдел технической поддержки по телефону рекомендуется находиться рядом с компьютером.

3. Установка

Система устанавливается с помощью инсталляционного набора.

 Установка должна осуществляться с правами суперпользователя.

3.1. Системные требования

Операционные системы:

- CentOS 7
- Ubuntu 20.04
- Ubuntu 22.04
- AstraLinux CE 2.12
- AstraLinux SE 1.7

При установке аддона осуществляется проверка на наличие в системе следующих маркеров совместимости:

- Наличие системной библиотеки GLIBC версии 2.17 или выше
- Наличие системных библиотек libgcc_s.so.1 и libstdc++.so.6
- Наличие менеджера системных служб systemctl

Программное обеспечение mono, которое используется при работе *PNTS Linux Training Addon*, входит в состав дистрибутива.

Минимальные аппаратные требования:

- Процессор Intel Core i5
- Видеокарта NVIDIA GTX 1070 (8 Гб) (рекомендуется NVIDIA GeForce GTX 1080 Ti)
- Оперативная память 16 Гб

Место на диске: 10 Гб.

Для работы утилиты конвертации моделей в формат CT2 требуется наличие Python 3.6.4-3.10 и следующих пакетов:

- numpy (поддерживаемый выбранной версией python)
- PyYAML версии 5.3.-7 для python
- CTranslate2 версии 2.14.0 для python

Требования к GPU

Для обучения моделей используются вычислительные возможности видеокарты (GPU).

В качестве GPU могут использоваться видеокарты NVIDIA. Минимальная модель видеокарты: NVIDIA GTX 1070.

Минимальное требование к памяти: 8 Гб (рекомендуется 11 Гб и больше).

 Чем больший объем памяти будет выделен, тем выше будет результат обучения модели.

Требуется драйвер видеокарты с поддержкой CUDA 11.

3.2. Установка продукта

PNTS Linux Training Addon функционирует совместно с программным продуктом *PNTS*. Для использования моделей, обученных с помощью PNTA, требуется наличие у пользователя PNTS с действующей лицензией.

Запустите установочный файл, используя следующую команду:

```
sudo chmod +x PNTS_Training_Addon_Uxxxx.run && sudo ./PNTS_Training_Addon_Uxxxx.run
```

где `PNTS_Training_Addon_Uxxxx.run` - название установочного файла *PNTS Linux Training Addon* (xxxx – номер версии).

Например, если установочный файл называется `PNTS_Training_Addon_U0017.run`:

```
sudo chmod +x PNTS_Training_Addon_U0017.run && sudo ./PNTS_Training_Addon_U0017.run
```

Процесс установки состоит из нескольких шагов, которые выполняются автоматически.

В процессе установки:

1. Ознакомьтесь с лицензионным соглашением и примите его условия.

    При отказе от условий лицензионного соглашения дальнейшая установка невозможна.

2. При необходимости укажите папку, в которую будет установлен программный продукт. По умолчанию: `/usr/local/ntaXX`.

   Например: `/usr/local/nta23`

После завершения установки в корневую папку будут скопированы файлы утилит, справка, дистрибутив mono, а также bash-скрипты для запуска процесса тюнинга (run_training.sh), получения BLEU-score (run_bleuscore.sh) и удаления продукта (uninstall/uninstall.sh).

3.3. Активация PNTS Linux Training Addon

В продукте реализована система лицензирования, позволяющая ограничивать доступ к основной функциональности продукта в зависимости от файла лицензии.

Вместе с набором устанавливается лицензионный файл *license.lic*, располагающийся в корневой папке набора. Данный лицензионный файл содержит информацию о статусе продукта (по умолчанию статус "Не активирован") и доступных языковых парах.

Чтобы активировать продукт, выполните следующее:

1. Получите идентификатор компьютера, привязанный к оборудованию системы, в которой установлен продукт. Для этого запустите исполнимый файл инсталлятора с ключом командной строки --id (-i).

   Например, если установочный файл называется `PNTS_Training_Addon_U0017.run`:

   ```
   sudo chmod +x PNTS_Training_Addon_U0017.run && ./PNTS_Training_Addon_U0017.run --id
   ```

   В результате выполнения данного файла в консоль выводится строка с идентификатором компьютера, например:

   ```
   Current hardware id: Y27wF82sEHHo25v7DUMqUMttqRhmALqzXt99qke7bZk=
   ```

2. Пришлите в службу поддержки компании ПРОМТ лицензионный файл и идентификатор компьютера `Current hardware id`.
3. Получив новый лицензионный файл, скопируйте его "поверх" файла, установленного с набором.

Результатом активации является новый лицензионный файл, который привязан к оборудованию и содержит статус "Активирован".

Утилиты при запуске проверяют наличие и валидность лицензионного файла. Если продукт активирован, то утилиты работают без ограничений. Если продукт не активирован, корректная работа аддона невозможна.

4. Общая схема работы

PNTS Linux Training Addon представляет собой набор утилит командной строки, которые последовательно запускаются с помощью bash-скриптов.

Скрипт **run_training.sh** запускает препроцессинг и обучение модели. Для оценки результатов обучения модели используется скрипт **run_bleuscore.sh**.

 Перед запуском bash-скриптов необходимо задать соответствующие параметры.

 Скрипты следует запускать из папки продукта (по умолчанию /usr/local/nta23/).

Общая схема работы может быть представлена в виде следующих шагов:

1. Подготовка исходных данных

• Подготовьте корпус параллельных текстов, на базе которых планируется провести обучение NMT-модели.

 Подробные сведения о подготовке исходных данных см. в разделе 5.

•Скопируйте модель, которую необходимо обучить, в любую папку.

 У пользователя, который запускает процесс настройки, должны быть права на запись в папку с моделью.

2. Настройка модели

• Задайте необходимые параметры в файле utils/settings.json и запустите **run_training.sh**, который автоматически выполнит препроцессинг текстового корпуса и непосредственно обучение модели.

 Подробные сведения о запуске процесса настройки модели см. в разделе 6.

3. Сохранение результатов настройки модели

• Установите настроенную NMT-модель в *PNTS*.

 Подробные сведения об установке моделей в PNTS вы найдете в руководстве администратора PNTS .

4. Оценка результатов настройки модели с помощью значения BLEU

 Для оценки результатов настройки убедитесь, что базовая модель подключена к одному профилю перевода (например, "Универсальный"), а настроенная модель подключена к другому профилю (например, "test").

a. Вычислите значение BLEU для базовой модели. Для этого задайте нужные параметры и запустите файл **run_bleuscore.sh**.

b. Вычислите значение BLEU для настроенной модели. Для этого задайте нужные параметры и запустите файл **run_bleuscore.sh**.

c. Сравните результаты BLEU, полученные в пп. a. и b. Если значение BLEU, полученное при переводе с настроенной моделью, больше, чем значение BLEU, полученное при переводе с базовой моделью, то результат настройки модели считается успешным.

 Подробные сведения об оценке результатов настройки модели см. в разделе 7.

 Подробную информацию об этапах настройки модели можно найти в следующих разделах данного документа.

5. Подготовка исходных данных

5.1. Текстовый корпус

Исходные данные (текстовый корпус) должны быть представлены либо в виде одного или нескольких tmx файлов, либо в виде набора параллельных текстовых файлов в кодировке UTF-8.

Текстовые файлы должны располагаться в одной папке и иметь расширения, соответствующие префиксам входного или выходного языков.

- Название текстовых файлов может быть произвольным (за исключением списка зарезервированных имен файлов). Необходимо, чтобы в папке имелась пара файлов с одним и тем же названием, но имеющих разные расширения, соответствующие префиксам входного и выходного языков.
- Путь к текстовому корпусу должен содержать только стандартную латиницу.
- Языковые префиксы см. в Приложении 1.

Минимальный рекомендуемый размер пользовательского корпуса - 5000 строк.

Если пользовательский корпус имеет менее 5000 примеров, то будет выведено предупреждение о том, что обучение на таком корпусе будет малоэффективным.

Список зарезервированных (служебных) имен файлов:

Условные обозначения:
[$src] – префикс входного языка
[$tgt] – префикс выходного языка

aligned-grow-diag-final.realigned	corpus.bpe.[$src]
corpus.bpe.[$src].sub	corpus.bpe.[$tgt]
corpus.bpe.[$tgt].sub	corpus.[$src]
corpus.[$src].o	corpus.full.bpe.[$src]
corpus.full.bpe.[$tgt]	corpus.full.bpe.with.basic.[$src]
corpus.full.bpe.with.basic.[$tgt]	corpus.full.bpe.[$src].dupl
corpus.full.bpe.[$tgt].dupl	corpus.[$tgt]
corpus.[$tgt].o	corpus.tok.def.clean.[$src]
corpus.tok.def.clean.[$tgt]	corpus.tok.def.[$src]
corpus.tok.def.[$tgt]	corpus.tok.[$src]
corpus.tok.[$src].sub	corpus.tok.[$tgt]
corpus.tok.[$tgt].sub	aligned-grow-diag-final.realigned.dupl
general.[$src].part	general.[$tgt].part
general.tok.def.[$src]	general.tok.def.[$tgt]

general.tok.def.clean.[$src]	general.tok.def.clean.[$tgt]
general.tok.[$src]	general.tok.[$tgt]
general.bpe.[$src]	general.bpe.[$tgt]
dev.bpe.[$src]	dev.bpe.[$src].output
dev.bpe.[$tgt]	dev.[$src]
dev.[$tgt]	dev.tok.[$src]
dev.tok.[$tgt]	test.[$src]
test.[$tgt]	

Таблица 5.1. Зарезервированные имена файлов

Примеры зарезервированных имен файлов для англо-русского направления:

aligned-grow-diag-final.realigned
corpus.bpe.en.sub
corpus.bpe.ru.sub
corpus.en.o
corpus.full.bpe.ru
corpus.ru.o
corpus.tok.def.clean.ru
corpus.tok.def.ru
corpus.tok.en.sub
corpus.tok.ru.sub
dev.bpe.en.output
dev.en
dev.tok.en
test.en
corpus.bpe.en
corpus.bpe.ru
corpus.en
corpus.full.bpe.en
corpus.ru
corpus.tok.def.clean.en
corpus.tok.def.en
corpus.tok.en
corpus.tok.ru
dev.bpe.en
dev.bpe.ru
dev.ru
dev.tok.ru
test.ru

Пример корпуса, состоящего из tmx-файла и нескольких текстовых файлов для англо-русского направления:

corpus.tmx
corpus1.en
corpus1.ru
corpus2.en
corpus2.ru

 У пользователя, который запускает процесс настройки, должны быть права на запись в папку с текстовым корпусом.

5.2. Подготовка базовой модели

В качестве исходных данных для настройки модели используется архив базовой модели. NMT-модели поставляются в виде архивов с расширением .zip.

Скопируйте архив модели, которую необходимо обучить, в любую папку.

 У пользователя, который запускает процесс настройки, должны быть права на запись в папку с архивом модели.

 Путь к архиву с базовой моделью необходимо указать в файле **settings.json**. Подробнее – см. раздел Запуск процесса настройки и задание параметров.

 Все указываемые пути к корпусам и моделям должны содержать только стандартную латиницу.

6. Настройка модели

6.1. Запуск процесса настройки и задание параметров

Процесс настройки модели запускается с помощью файла **run_training.sh** из каталога установленного приложения.

> Запускайте *run_training.sh* из корневой директории продукта (по умолчанию /usr/local/nta23/).Если запуск *run_training.sh* должен быть осуществлен с правами суперпользователя, то вызовите команду `sudo ./run_training.sh` из корневой директории продукта.

Перед запуском **run_training.sh** отредактируйте файл **settings.json** (по умолчанию: **utils/settings.json**), задав параметры настройки модели через соответствующие переменные.

Параметры:

Source_Language	Префикс входного языка.
Target_Language	Префикс выходного языка.
Corpus	Путь к папке с пользовательским корпусом. Путь должен содержать только стандартную латиницу.
Validation_Percent	Суммарный объем корпуса для валидации и тестового корпуса в процентах от исходного корпуса. При этом исходный обучающий корпус уменьшается на соответствующее количество строк. Рекомендуется задавать данный параметр в пределах от 1 до 10%. Суммарное максимальное число строк, выделяемое для двух этих корпусов, не может превышать 6000.
Baseline_Model	Путь к архиву с базовой моделью. Путь должен содержать только стандартную латиницу.
Result_Model	Путь к архиву с результирующей (настроенной) моделью. Проверьте, что папка, указанная для сохранения модели, существует.
Model_Workspace	Зарезервированный размер памяти (МБ), отведенный под процесс настройки модели. Если в качестве значения указано "auto", этот размер вычисляется перед запуском настройки модели, исходя из доступной памяти

	GPU устройства и из размера базовой модели. **Model_Workspace** не включает в себя память, отводимую под загрузку модели и работу marian. Рекомендуется подбирать данный параметр максимально большим, оставляя при этом свободными ~ 2.5 Гб на GPU устройстве. Если выделенный объем памяти слишком большой, процесс обучения будет остановлен с сообщением об ошибке.
Model_Valid_Freq	Частота валидаций настраиваемой модели в процессе настройки. Указывается число обновлений (updates), после которого необходимо запустить валидацию. Если в качестве значения указано "auto", значение выбирается автоматически (за весь процесс настройки будет выполнено 3 валидации). Порядок величин зависит от объема обучающего корпуса, выбранного значения параметра **Model_Workspace** и желаемого числа валидаций на весь процесс обучения. Рекомендуется подбирать данный параметр так, чтобы за весь процесс обучения выполнились 3-4 валидации. Большее число валидаций приводит к неоправданному увеличению времени обучения модели.
Model_GPU_Devices	Используемая видеокарта. Идентификатор GPU, используемого при обучении. Все GPU устройства, подключенные к компьютеру, получают числовые идентификаторы от 0 до N-1, где N – число GPU устройств на машине. Если указанный идентификатор выходит за границы диапазона, процесс обучения будет остановлен с сообщением об ошибке.
Training_Set_Min_Size	Минимальный размер, до которого дублируется пользовательский корпус. По умолчанию: 1 000 000 строк.
Python_path	Установленный Python (например, "python3.7"). Версия Python должна соответствовать системным требованиям и для нее должны быть установлены соответствующие пакеты (см. раздел 3.1).
CT2	Автоматическая конвертация тренируемой модели marian в CT2. Если конвертация в CT2 необходима, укажите значение "1". Если для данного параметра указано значение "0", конвертация

	производиться не будет. По умолчанию конвертация включена.

Таблица 6.1. Параметры файла settings.json

Если предполагается работать на системе с несколькими GPU, для выбора определенного устройства через параметр *Model_GPU_Devices* вызовите утилиту *promt-gpuinfo.exe*, которая находится в папке utils установленного аддона. Пример вызова утилиты при установке в папку по умолчанию:

```
/usr/local/nta23/utils/promt-gpuinfo.exe
```

Первое выведенное в результатах устройство имеет индекс "0".

Для обучения моделей следует выбирать только одно устройство GPU.

Для конвертации настраиваемых моделей marian в CT2 необходимым условием является наличие Python и установленных для него пакетов. Подробнее об установке Python на разных системах см. в Приложении 2.

Все указываемые пути к корпусам и моделям должны содержать только стандартную латиницу.

Языковые префиксы см. в Приложении 1.

Пример файла settings.json для настройки англо-русской модели:

```
{
  "Source_Language": "en",
  "Target_Language": "ru",
  "Corpus": "/home/admin1/PNTA23/tmx/RU",
  "Baseline_Model": "/home/admin1/PNTA23/models/er_23.1_09.22.zip",
  "Model_Workspace": "auto",
  "Model_CPU_Threads": "0",
  "Model_GPU_Devices": "0",
  "Validation_Percent": "10",
  "Model_Valid_Freq": "auto",
  "Result_Model":
"/home/admin1/PNTA23/models/trained/eg_23.1_09.22_trained_ct2_GPU.zip",
  "Training_Set_Min_Size": "1000000",
  "Python_Path":"python3",
  "CT2":"1"
}
```

6.2. Этапы обучения модели

Весь процесс обучения модели можно разбить на два этапа: препроцессинг текстового корпуса и непосредственно обучение модели.

6.2.1. Препроцессинг

Препроцессинг состоит из следующих шагов:

1. **Создание исходного параллельного текстового корпуса по папке с входными файлами**

   Выполняется утилитой **promt.nts.extractcorpus.exe**. На вход утилите передается путь к папке с текстовым корпусом. На выходе получается пара файлов corpus.$src и corpus.$tgt, в которых объединены все текстовые сегменты для входа и выхода из исходной папки.

2. **Нормализация пунктуации и токенизация пользовательского корпуса**

   Производится утилитой **promt.nts.tokenize**. На вход подается корпус corpus, на выходе получается токенизированный корпус corpus.tok.

3. **Прочистка пользовательского корпуса**

   Производится утилитой **promt.nts.cleancorpus**. Из сета corpus исключаются пустые, дублирующиеся или слишком длинные строки. Максимальная длина строки на входе и выходе не должна превышать 100 слов и 1000 символов.

   Кроме этого, утилита *promt.nts.cleancorpus.exe* в процессе работы отфильтровывает и выводит в отдельный лог *corpus.tok.def.clean.badstring.txt* строки, которые не могут быть использованы для обучения модели. Лог сохраняется в папке с пользовательским корпусом.

   В логе указывается причина фильтрации, входная строка и выходная строка.

   Например:

```
Bad length:
Sberbank 's digital solutions are among the most widely used in the world .
компаний
```

   Ниже приведены причины, по которым строки могут попасть в лог:

Сообщение в логе	**Причина попадания в лог**
Duplicate	Пара вход-выход уже встречалась ранее
Equal lines	Вход = выход
Bad length	Вход или выход имеет меньше 1 слова (пустой), больше 100 слов, либо менее 10 символов

Bad alphanumeric % in source/target	Во входной/ выходной строке кол-во букв < 30% от всего входа
Source/Target is empty	После выполнения обработки во входной/выходной строке не осталось символов
Source or target is too long or too short	После выполнения обработки во входной/выходной строке стало меньше 1 или больше 100 слов
Bad source length/target length ratio	После выполнения обработки отношение длин входного текста к выходному > 9 или наоборот, если отношение выходного к входному >9.

4. Создание корпуса для валидации данных и тестового корпуса

Выполняется утилитой **promt.nts.extractrandomset**. По заданному проценту извлекаемых сегментов (Validation_Percent) утилита выбирает случайные сегменты из исходного корпуса. Максимальное число строк в случайной выборке ограничено значением 6000. Полученная случайная выборка строк равномерно распределяется между создаваемыми новыми корпусами: корпусом для валидации (dev-сет) и тестовым корпусом (test-сет). Результат записывается в файлы dev.tok.$src, dev.tok.$tgt и test.tok.$src, test.tok.$tgt.

Файлы dev.tok.$src и dev.tok.$tgt используются для валидации данных в процессе настройки модели.

Для тестового токенизированного корпуса test.tok выполняется операция детокинезации, результат записывается в файлы test.$src и test.$tgt.

Файлы test.$src и test.$tgt используются для оценки BLEU после завершения настройки модели.

Из исходного корпуса удаляются строки, попавшие в случайную выборку, т.о. обучение происходит на исходном корпусе минус корпус для валидации и тестовый корпус.

5. Генерация синтетических данных с незнакомыми словами для пользовательского корпуса

Производится утилитой **promt.nts.makeunkdata**. По сету corpus строится дополнительный сет, содержащий пометки <unk> вместо случайно взятых слов. Дополнительный сет записывается в файлы с расширением .sub.

6. Токенизация тестовых, валидационных и синтетических данных

Производится утилитой **promt.nts.tokenize**. На выходе получаются обработанные файлы основного сета, синтетического сета и сета dev с дополнительной строкой .bpe. в названии.

7. Построение выравнивания для пользовательских данных

Выполняется утилитой **promt.nts.makeunkdata**. На вход подается объединенный корпус, результат работы записывается в файл aligned-grow-diag-final.realigned.

8. **Подсчет количества необходимых итераций для обучения**.

   Осуществляется по формуле 1000000/кол-во пользовательских данных +1.

6.2.2. Обучение модели

После подготовки и обработки исходных данных запускается непосредственно процесс обучения (настройки) NMT-модели с помощью приложения командной строки **marian.exe**.

Основные параметры вызова приложения (путь и название настраиваемой модели) задаются в файле **settings.json**.

Непосредственно запуск **marian.exe** выполняется утилитой **promt.nts.MarianLauncher.exe**.

Утилита **promt.nts.MarianLauncher.exe**:

1. Распаковывает архив с базовой моделью во временную папку. Временная папка создается в той же папке, где находится базовая модель.

   После завершения обучения временная папка удаляется.

2. Подготавливает распакованные файлы архива к последующей настройке модели:

   a. Находит файл модели "model.npz.best-bleu.npz" и переименовывает в "model.baseline.npz";

   b. Находит файл настроек модели "model.npz.best-bleu.npz.decoder.yml", устанавливает в нем значение "aligment: 0", переименовывает в "model.baseline.npz.decoder.yml".

3. Вычисляет значения параметра dim-vocab, анализируя файлы vocab.src.yml и vocab.tgt.yml;

4. При необходимости автоматически вычисляет значения параметров Model_Workspace и Model_Valid_Freq.

5. После успешного завершения обучения модели формирует файл архива с результирующей моделью и записывает его по пути, указанном в параметре Result_Model.

   После завершения обучения происходит конвертация модели в ct2, если соответствующий параметр указан в файле конфигурации *settings.json*. Конвертация выполняется утилитой promt.nts.ct2converter.exe.

В ходе выполнения настройки создаются файлы с логами:

- train.log – вывод информации о ходе обучения;
- valid.log – информация о валидациях.

  Для конвертации настраиваемых моделей marian в CT2 необходимым условием является наличие Python и установленных для него пакетов.

Обученную NMT-модель можно подгрузить в PNTS средствами, предусмотренными в установленной версии PNTS. Подробнее об установке NMT-моделей см. руководство администратора PNTS.

7. Вычисление финального значения BLEU

Подсчет финального значения BLEU производится с помощью утилиты **promt.nts.BSUtils.exe**, запускаемой командным файлом **run_bleuscore.sh**.

 Скрипт следует запускать из папки продукта (по умолчанию /usr/local/nta23/).

Утилита вычисляет значение BLEU, сравнивая эталонный перевод с результатом машинного перевода исходного текста, выполненного сервером перевода *PNTS* с заданным профилем перевода.

Для оценки результатов настройки модели выполните следующее:

1. Вычислите значение BLEU для базовой модели.

   Для этого запустите файл **run_bleuscore.sh** с нужными параметрами и получите значение BLEU, сравнивая эталонный перевод (test.$tgt из test-сета) с результатом машинного перевода исходного текста (test.$src из test-сета), выполненного с профилем перевода, в котором подключена базовая модель.

2. Вычислите значение BLEU для настроенной модели.

   Для этого запустите файл **run_bleuscore.sh** с нужными параметрами и получите значение BLEU, сравнивая эталонный перевод (test.$tgt из test-сета) с результатом машинного перевода исходного текста (test.$src из test-сета), выполненного с профилем перевода, в котором подключена настроенная модель.

3. Сравните полученные результаты BLEU.

   Если значение BLEU, полученное при переводе с настроенной моделью, больше, чем значение BLEU, полученное при переводе с базовой моделью, то результат настройки модели считается успешным.

Значение BLEU выдается как результат работы утилиты в консоль.

7.1. Параметры для запуска утилиты

Для запуска утилиты в командном файле **run_bleuscore.sh** необходимо задать ряд параметров:

- **PTS_URL** – строка с URL сервера перевода (например, PTS_URL=http://localhost/pnts);
- **PTS_LINUX** – использование Windows- или Linux-версии сервера перевода. Значения:

  0 – используется Windows-версия сервера перевода PNTS;

  23 – используется PNTS Linux версии 23 (например, PNTS23.1 - в таком случае параметр будет выглядеть так: PTS_LINUX=23);

  22 –- используется PNTS Linux версии 22 (например, PNTS22.1 - в таком случае параметр будет выглядеть так: PTS_LINUX=22).
- **PTS_USER** – пользователь PNTS (для случая Forms аутентификации);

- **PTS_USER_PASSWORD** - пароль пользователя PNTS (для случая Forms аутентификации);
- **SRC** – префикс исходного языка;
- **TGT** – префикс языка перевода;
- **TOPIC** – профиль перевода, с которым будет производится перевод;
- **SRC_FILE** – путь к исходному текстовому файлу test.$src из test-сета (кодировка UTF-8).
- **REF_FILE –** путь к текстовому файлу с эталонным переводом test.$tgt из test-сета (кодировка UTF-8).

 Во время подсчета BLEU для базовой и обученной NMT-моделей не рекомендуется изменять конфигурацию компьютера и/или настройки сервера перевода.

 Языковые префиксы см. в Приложении 1.

Пример файла **run_bleuscore.sh** для вычисления значения BLEU для англо-русской модели, подключенной в профиле "test":

```
#!/bin/bash

PTS_URL=http://localhost/pnts
PTS_LINUX=23
PTS_USER=
PTS_USER_PASSWORD=

SRC=en
TGT=ru
TOPIC=test
REF_FILE=/home/user/promt-nts/corpus/test.ru
SRC_FILE=/home/user/promt-nts/corpus/test.en

./mono/mono ./utils/promt.nts.bsutil.exe --ref="$REF_FILE" --sf="$SRC_FILE" --
url=$PTS_URL "--topic=$TOPIC" --src=$SRC --tgt=$TGT --linux=$PTS_LINUX --
user="$PTS_USER" --password="$PTS_USER_PASSWORD" --tokenizer="promt.nts.tokenize.exe"
```

Глоссарий

BLEU (Bilingual Evaluation Understudy) — измерение различий между автоматическим переводом и одним или несколькими эталонными пользовательскими переводами одного исходного предложения. Алгоритм BLEU сравнивает последовательные фразы автоматического перевода с последовательными фразами, которые он находит в эталонном переводе, и взвешенно подсчитывает количество совпадений. Эти совпадения не зависят от позиции. Высшая степень совпадения указывает на более высокую степень сходства с эталонным переводом и более высокий балл. Внятность и грамматика не учитываются.

Апдейт (update) – обновление весов при обучении.

Валидация – перевод dev-сета с вычислением BLEU.

Корпус для валидации (dev-сет) – из входного корпуса после токенизации и очистки случайным образом выбираются строки (заданный процент от общего числа, максимальное количество – 6000 строк). Из входного корпуса выбранные строки удаляются. Половина из выбранных строк записываются в новый корпус, называемый **корпусом для валидации** (dev.tok.$src, dev.tok.$tgt). Вторая половина записывается в *тестовый корпус*. Модель обучается на строках "входной корпус минус корпус для валидации и минус тестовый корпус". После каждого цикла обучения делается перевод корпуса для валидации и считаются показатели эффективности обучения (BLEU и др.), которые оцениваются после каждого цикла, и, если их прирост значительно замедляется или останавливается, принимается решение об окончании обучения.

Тестовый корпус для подсчета финального BLEU (test-сет) – из входного корпуса после токенизации и очистки случайным образом выбираются строки (заданный процент от общего числа, максимальное количество – 6000 строк). Из входного корпуса выбранные строки удаляются. Половина из выбранных строк записываются в *корпус для валидации*. Вторая половина записывается в новый корпус, называемый **тестовым корпусом** (test.$src, test.$tgt). После завершения обучения делается перевод тестового корпуса базовой моделью и настроенной моделью и вычисляются BLEU для каждой модели.

Минибатч – количество предложений, которое просматривает модель при обучении за один апдейт.

Модель нейронного машинного перевода (NMT-модель) – лингвистические данные, необходимые для выполнения нейронного машинного перевода (NMT).

Нормализация пунктуации – знаки пунктуации приводятся к последовательности &<code>; (экранирование).

Построение выравнивания – токены из входной строки соотносятся с токенами из выходной строки, и результат записывается в формате, понятном для marian (например, 1-1 2-4 и т.д.).

Профиль перевода – совокупность сохраненных лингвистических настроек, которые доступны во всех приложениях переводчика PROMT и позволяют настраивать его на перевод текстов определенной предметной области. В некоторых продуктах PROMT называется также тематикой.

Сегментация BPE и обработка регистра – входные данные приводятся в формат, с которым работает marian; сегментация BPE - процесс разбиения слов на сегменты по словарю BPE,

обработка регистра - приведение к нижнему регистру с сохранением информации о начальном состоянии (например, |C - начиналось с верхнего регистра, |U - было полностью в верхнем регистре).

Токен – слово или знак пунктуации, обрамленный пробелами.

Токенизация – разделение строки на токены, которыми являются слова, знаки пунктуации и неалфавитные символы. Все токены отделяются друг от друга символом пробела.

Эпоха – просмотр во время обучения всего корпуса.

Языковая пара – указывает, с какого языка и на какой будет переводиться текст.

Приложение 1. Префиксы для обозначения языков

В PNTS для обозначения языков используются префиксы в формате RFC. В таблице ниже приведены префиксы языков, которые используются при запуске обучения моделей, а также при запуске утилиты для подсчета BLEU.

> Для моделей с китайским языком при подсчете BLEU следует использовать префиксы zh-tw и zh-cn, а для запуска обучения моделей – префикс zh.

язык	префикс
Азербайджанский	az
Английский	en
Арабский	ar
Армянский	hy
Белорусский	be
Болгарский	bg
Венгерский	hu
Вьетнамский	vi
Греческий	el
Грузинский	ka
Датский	da
Иврит	he
Индонезийский	id
Испанский	es
Итальянский	it
Казахский	kk
Каталанский	ca
Киргизский	ky
Китайский (традиционный)	zh-tw
Китайский (упрощенный)	zh-cn
Корейский	ko
Латышский	lv
Литовский	lt
Малайский	ms
Мальтийский	mt
Немецкий	de
Нидерландский	nl

Норвежский	no
Нюнорск	nn
Польский	pl
Португальский	pt
Румынский	ro
Русский	ru
Сербский	sr
Словацкий	sk
Словенский	sl
Таджикский	tg
Тайский	th
Тамильский	ta
Татарский	tt
Турецкий	tr
Туркменский	tk
Узбекский	uz
Украинский	uk
Урду	ur
Фарси	fa
Филиппинский	fil
Финский	fi
Французский	fr
Хинди	hi
Хорватский	hr
Чешский	cs
Шведский	sv
Эстонский	et
Японский	ja

Приложение 2. Установка Python на системах Linux

Для всех поддерживаемых систем, кроме AstraLinux 1.7 SE, предусмотрена только онлайн-установка Python и необходимых пакетов. Для AstraLinux 1.7 SE установка необходимых компонентов может производиться как онлайн, так и из папки Prerequisites в составе набора.

CentOS 7

1. Убедитесь, что подключен репозиторий epel:

   ```
   yum repolist
   ```

   Если репозиторий подключен, будет выведена следующая строка:

   ```
   epel/x86_64              Extra Packages for Enterprise Linux 7 - x86_ 13 770
   ```

2. Если репозиторий не подключен, подключите его:

   ```
   sudo yum install epel-release
   ```

3. Установите Python 3.6:

   ```
   sudo yum install python36
   ```

   После успешной установки к данной версии python можно обращаться через команды "python3" или " python3.6".

   Проверить установленную версию python можно одной из следующих команд:

   ```
   python3 –V
   python3.6 –V
   ```

4. Обновите pip:

   ```
   sudo python3 -m pip install --upgrade pip
   ```

5. Установите необходимые пакеты:

   ```
   sudo python3 -m pip install ctranslate2==2.14.0
   ```

   При установке пакета ctranslate2 будут автоматически установлены подходящие версии пакетов PyYAML и numpy.

Ubuntu 20.04/22.04

Python включен в дистрибутив Ubuntu 20.02/22.04, поэтому установка самого Python на данной ОС не требуется. Необходимо установить pip и пакеты для Pyton.

Для этого:

1. Обновите список пакетов

```
sudo apt update
```

2. Установите pip

```
sudo apt install python3-pip
```

3. Установите пакеты

```
sudo python3 -m pip install ctranslate2==2.14.0
```

 При установке пакета ctranslate2 будут автоматически установлены подходящие версии пакетов PyYAML и numpy.

Astra Linux 2.12 CE

1. Обновите список пакетов

```
sudo apt update
```

2. Установите python3.7

```
sudo apt install python3.7
```

3. Установите pip

```
sudo apt install python3-pip
```

4. Обновите pip

```
sudo python3.7 -m pip install --upgrade pip
```

5. Установите пакеты

```
sudo python3.7 -m pip install ctranslate2==2.14.0
```

 При установке пакета ctranslate2 будут автоматически установлены подходящие версии пакетов PyYAML и numpy.

Astra Linux 1.7 SE

Python3.7 включен в дистрибутив Astra Linux 1.7 SE, в связи с чем установка самого Python на данной ОС не требуется. Необходимо установить pip и пакеты для Pyton. Установка может производиться как онлайн, так и из папки Prerequisites в составе набора.

Онлайн-установка:

1. Обновите список пакетов

```
sudo apt update
```

2. Установите pip

```
sudo apt install python3-pip
```

3. Обновите pip

```
sudo python3.7 -m pip install --upgrade pip
```

4. Установите пакеты

```
sudo python3.7 -m pip install ctranslate2==2.14.0
```

При установке пакета ctranslate2 будут автоматически установлены подходящие версии пакетов PyYAML и numpy.

Офлайн-установка:

В случае работы в системе без интернет-подключения, установку необходимых компонентов можно осуществить в режиме офлайн.

Офлайн-установка выполняется из папки Prerequisites (по умолчанию */usr/local/nta23/Prerequisites/python3*).

Для установки необходимых компонентов выполните следующую последовательность действий:

1. Установите pip

```
sudo python3 ./pip-23.0.1-py3-none-any.whl/pip install ./pip-23.0.1-py3-none-any.whl
```

2. Установите необходимые пакеты:

- numpy

```
sudo python3 -m pip install ./Astra\ SE\ 1.7/numpy-1.21.6-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl
```

- PyYAML

```
sudo python3 -m pip install ./Astra\ SE\ 1.7/PyYAML-6.0-cp37-cp37m-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_6
4.whl
```

- ctranslate2

```
sudo python3 -m pip install ./Astra\ SE\ 1.7/ctranslate2-2.14.0-cp37-cp37m-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl
```