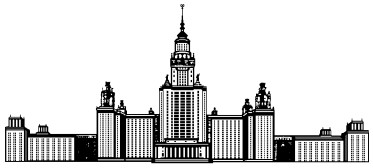

Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Скачков Николай Андреевич

СОВМЕСТНОЕ ОБУЧЕНИЕ ПРЯМОЙ И ОБРАТНОЙ МОДЕЛИ МАШИННОГО ПЕРЕВОДА
JOINT TRAINING OF DIRECT AND REVERSE MACHINE TRANSLATION MODELS

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:
д.ф-м.н., доцент
К.В.Воронцов

Москва, 2021

Аннотация

Машинный перевод является активно развивающейся областью обработки текстов естественного языка, ставящей своей целью автоматический перевод текста с одного языка на другой. Существующие методы машинного перевода обладают высоким качеством для некоторых языков, однако задача перевода, в целом, не решена. Существуют различные методы улучшения качества систем перевода. В этой работе основное внимание уделено методам, направленным на улучшения консистентности переводов моделей, соответствующих прямым и обратным им языковым направлениям. Будут рассмотрены уже существующие методы, а также возможности их улучшения на практике. Также будет представлена попытка дать общее теоретическое обоснование для таких методов с точки зрения вероятностного моделирования.

Содержание

1 Вступление **3**

2 История **4**

3 Описание подхода **6**

3.1 Задача машинного перевода 6

3.2 Использование обратной модели 6

4 Используемые методы **10**

4.1 Архитектура модели 10

4.2 Оценка качества перевода 13

4.3 Оценка качества циклического перевода 14

5 Эксперименты **15**

5.1 Детали экспериментов 15

5.2 Эксперименты с обучением с нуля 17

5.3 Эксперименты с доучиванием моделей 18

6 Заключение **22**

1 Вступление

Задача машинного перевода является одной из важнейших задач анализа текстов естественного языка. Перевод текстов является достаточно трудоёмким процессом, требующим существенное количество человеческого труда и времени.

Современные модели перевода используют нейросетевые модели, обученные на параллельных предложениях [1]. Эти модели на основе большого количества параллельных текстов выучивают межъязыковые закономерности и используют их при переводе. При этом в момент обучения модель учится повторять те обучающие тексты, которые подаются ей на вход.

Важным критерием при разработке систем машинного перевода является согласованность результатов, полученных с помощью различных способов перевода. Так, например, неожиданно, переводя текст с английского на русский, получить при переводе результата обратно на английский текст, значительно отличающийся от исходного. К сожалению, такие ситуации могут возникать, так как прямая и обратная модели учатся на различных данных, ориентированных на целевой язык в каждом случае.

Для борьбы с такого рода несогласованностями был предложен метод совместного обучения прямой и обратной моделей перевода [2]. Более того, такое обучение позволяло улучшить качество перевода, в целом, так как сигнал от обратной модели помогал улучшать перевод прямой и наоборот.

Проблема такого обучения заключается в необходимости одновременно оптимизировать параметры прямой и обратной моделей на протяжении всего обучения, что из-за ограничений памяти на вычислительных устройствах на практике приводит к более долгому обучению. В оригинальной работе были рассмотрены рекуррентные модели перевода [3], но для современных архитектур такие органичения приводят к неприменимости метода на практике. Более того, как будет показано в данной работе, в начале обучения сигнал от обратной модели оказывается крайне шумным, так что авторам приходилось оценивать полезность сигнала с помощью дополнительных языковых моделей, что требует еще больше вычислительных ресурсов и времени.

Так, в данной работе будет представлен способ решения проблемы с длительным обучением методом оптимизации только прямой модели и инициализацией весов из уже пре-

добученной модели. Также в работе будут продемонстрированы причины неприменимости предыдущего подхода на практике для современных архитектур машинного перевода. Более того, в работе представлены теоретические обоснования предложенной модели, которые довольно поверхностно описаны в первоначальном исследовании.

2 История

Задача машинного перевода получила распространение в середине 20-го века. Первыми системами машинного перевода были статистические и основанными на ручных правилах. Данные системы были низкого качества, однако передавали смысл и использовались в качестве подстрочника. Долгое время область развивалась медленно, так как не было понимания, как улучшать качество подобных систем.

Существенное распространение системы автоматического перевода получили после возникновения статистического машинного перевода (SMT) [4] в 90-ые годы 20-го века. Идея данного подхода заключается в переводе каждого слова с помощью частотного словаря, собранного по параллельным корпусам текстов. Данная идея развилась во фразовый машинный перевод (PBMT) [5], где переводились уже не отдельные слова, а фразы, с помощью фразовых таблиц, собранных по параллельным корпусам. Данный способ перевода лучше справлялся с устойчивыми выражениями и существенно лучше передавал смысл текстов. Первые онлайн-системы перевода были основаны именно на технологии PBMT.

Однако самый значимый прорыв был получен после использования нейросетевых технологий для перевода [1]. Данный способ позволял выучивать в модели нетривиальные языковые зависимости и использовать их при переводе. Некоторые переводы стали неотличимы по качеству от человеческих.

К сожалению, нейросетевой перевод также использует параллельные тексты при обучении как и PBMT. Следовательно, ошибки в данных быстро выучиваются и модель начинает повторять ошибки, которые видела. Подготовить очень большой обучающий корпус без ошибок практически невозможно, поэтому нейросетевые модели страдают от чрезмерных переводов, недопереводов, ошибок согласования. Каждая из этих ошибок появляется из-за наличия в данных не совсем параллельных предложений, так называемых ошибок

выравнивания, а также из-за грамматических и лексических ошибок в текстах.

Существует множество способов улучшения качества автоматического нейросетевого перевода. Некоторые подходы улучшает качество с помощью улучшения консистентности переводов. Так, в статье [2] предлагают во время обучения делать переводы моделями и использовать увеличивать схожесть обратного перевода с исходным текстом. Такой способ заставляет переводить более точно, не давая, в идеале, модели добавлять или терять смысл исходного предложения при переводе. Исследование применимости и улучшение данного подхода и является целью данной работы.

3 Описание подхода

В данном разделе будут описаны теоретические основы для существующих методов обучения моделей машинного перевода, а также для предлагаемого метода.

3.1 Задача машинного перевода

В задаче машинного перевода задан набор пар предложений на двух языках $\{(x_i, y_i)\}_{i=1}^N$. Будем считать, что x_i принадлежат языку входа (ЯВ), а предложения y_i — целевому языку (ЦЯ). Требуется максимизировать правдоподобие переводов, при условии входных предложений, что при логарифмировании выглядит следующим образом:

$$\sum_{i=1}^{N} \log P_\theta(y_i|x_i) \longrightarrow \max_\theta,$$

где θ — параметры модели.

При этом требуется, чтобы модель могла не просто оценивать вероятность перевода, но и позволяла бы генерировать перевод, то есть была бы генеративной. Существует несколько способов это сделать. Все эти способы различаются в способе представления $\log P_\theta(y|x)$ с точки зрения модели. Наиболее распространёнными являются авторегрессионные модели [1], которые генерируют перевод пословно, итеративно слева-направо. С точки зрения правдоподобия это выглядит следующим образом:

$$\log P_\theta(y|x) = \sum_{t=1}^{|y|} \log P_\theta(y^t|y^{<t}, x), \tag{1}$$

где y^t – это t-ое слово перевода, а $y^{<t}$ – это префикс перевода до t-го слова. Хоть ограничение авторегрессионности и является достаточно сильным, однако оно позволяет эффективно итеративно генерировать переводы, избегая полного перебора всевозможных переводов. Именно такой способ перевода наиболее распространён, и далее мы будем предполагать все модели в данной работе авторегрессионными.

3.2 Использование обратной модели

Рассматриваем конструкцию циклических переводов ЯВ→ ЦЯ→ ЯВ. То есть предложение на языке входа переводится на целевой язык с помощью прямой модели и обратно с

помощью обратной. Необходимо формализовать требование о том, что такой циклический перевод должен совпадать с исходным текстом.

Для этого введём следующую конструкцию, показывающую насколько вероятно получить текст x' при циклическом переводе текста x:

$$P_{\text{cycle}}(x'|x) = \mathbb{E}_{y \sim P_\Theta(y|x)} P'(x'|y), \tag{2}$$

где $P_\Theta(y|x)$ — параметризованная прямая модель перевода, а $P'(x|y)$ — обратная модель.

Данное выражение достаточно интуитивно, однако вычислить точное значение вероятности не представляется возможным с вычислительной точки зрения, так как математическое ожидание берётся по всевозможным текстам y целевого языка, а их число бесконечно. Для приближенного вычисления, например для валидации обучаемого алгоритма можно воспользоваться процедурой выбора по значимости, то есть оценкой математического ожидания значением на одном примере, сэмплированном из распределения $P_\Theta(y|x)$.

Теперь перейдём к формулировке функции потерь, с помощью которой будут оптимизироваться параметры модели перевода. Наше требование заключается в увеличение вероятности исходного текста при циклическом переводе, то есть, формулируя как задачу максимизации логарифма правдоподобия:

$$\log P_{\text{cycle}}(x|x) = \log \mathbb{E}_{y \sim P_\Theta(y|x)} P'(x|y) \longrightarrow \max_\Theta$$

Данное выражение нам нужно будет оптимизировать методом градиентного подъема, так как разнообразие методов используемых для обучения нейронных сетей достаточно ограничено. Перед подсчётом градиента заметим, что взятие логарифма, необходимое для стабильности методов обучения, усугубило ситуацию. Теперь выражение нельзя несмещённо оценить одним сэмплом. Поэтому воспользуемся неравенством Йенсена для вогнутых функций. Так как логарифм — вогнутая функция, а вероятностное распределение образует выпуклую комбинацию, получим:

$$\log \mathbb{E}_{y \sim P_\Theta(y|x)} P'(x|y) \geq \mathbb{E}_{y \sim P_\Theta(y|x)} \log P'(x|y) =: L(x)$$

Далее будем максимизировать именно полученную оценку $L(x)$. Так как оценка нижняя, ее максимизация означает максимизацию исходной функции потерь.

Посчитаем градиент полученной оценки. Именно он нам необходим для обучения модели. Для этого распишем математическое ожидание как интеграл, и внесём градиент внутрь интеграла:

$$\nabla_\Theta L(x) = \nabla_\Theta \int P_\Theta(y|x) \log P'(x|y) dy =$$
$$= \int \log P'(x|y) \nabla_\Theta P_\Theta(y|x) dy$$

Полученная запись градиента оценки не является более математическим ожиданием по распределению, так как внутри интеграла нет вероятности, поэтому данное выражение нельзя оценить с помощью сэмплирования. Для решения этой проблемы воспользуемся приемом:

$$\frac{\partial \log f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$$

Тогда оценку можно переписать следующим образом:

$$\nabla_\Theta L(x) = \int \log P'(x|y) P_\Theta(y|x) \nabla_\Theta \log P_\Theta(y|x) dy =$$
$$= \mathbb{E}_{y \sim P_\Theta(y|x)} \log P'(x|y) \nabla_\Theta \log P_\Theta(y|x)$$

Мы смогли представить градиент нижней оценки в виде математического ожидания некой случайной величины. Это математическое ожидание все еще невычислимо, так как интегрирование ведётся по бесконечному множеству. Но теперь мы можем несмещенно оценить это математическое ожидание значением на одном сэмпле, в соответствии с процедурой выбора по значимости:

$$\nabla_\Theta L(x) \approx \log P'(x|y) \nabla_\Theta \log P_\Theta(y|x), \quad y \sim P_\Theta(y|x) \tag{3}$$

Полученная несмещенная оценка нижней оценки функции потерь имеет смысл взвешенных вероятностей прямой модели с помощью вероятностей обратной модели. То есть, если обозначить за $\mathcal{L}(x, y)$ — функцию потерь логарифма правдоподобия прямой модели для пары предложений (x, y), то имеют место следующие формулы для вычисления

градиентов:

$$\nabla_\Theta \mathcal{L}(x, y) = \nabla_\Theta \log P_\Theta(y|x)$$
$$\nabla_\Theta L(x) \approx w(x, y) \nabla_\Theta \log P_\Theta(y|x), \quad y \sim P_\Theta(y|x),\ w = \log P'(x|y)$$

Запись функции потерь в виде взвешенной суммы даёт некоторую интерпретацию. Можно заметить, что чем вероятнее получить исходный текст с помощью обратного перевода из y, тем больше модель будет поощрять прямую модель переводить x как y.

При внешнем сходстве функций потерь обычной модели перевода и модели с циклическими переводами, в последней никак не используются реальные тексты из целевого языка. Модель сама генерирует текст перевода и увеличивает вероятность такого перевода исходного текста только если у него достаточно высокая вероятность обратного перевода в исходное предложение.

Стоит отметить, что полученные выражения совпали с формулами из оригинальной работы [2], однако её авторы предложили эти формулы как данность. В данной же работе предложена исходная модель, дающее такие выражения для оптимизации, что даёт пространство для улучшения данного подхода.

4 Используемые методы

4.1 Архитектура модели

В качестве архитектуры для авторегрессионного перевода лучшее качество показывает архитектура Transformer [6]. Модель представляет из себя кодировщик и декодеровщик, каждый из которых состоит из повторенных несколько раз одинаковых блоков. Каждый блок одинаково применяется ко всем токенам. При этом в кодировщике, в процессе обработки слова доступна информация обо всех словах из контекста, а в декодировщике — только к токенам, предшествующим данному.

Каждый блок в кодировщике Transformer состоит из слоя внимания и двухслойной полносвязанной нейросети. При этом изначально входные слова проецируются в пространство эмбеддингов и к ним добавляется вектор, кодирующий позицию слова в тексте. С помощью этого кодирования у модели есть возможность на каждом слое узнавать позицию токена в тексте. Общая архитектура сети представлена на Рис. 1.

Рассмотрим теперь подробнее как устроен механизм внимания в архитектуре Transformer. Пусть h_i — это векторное представление i-го слова на каком-то слое нейросети. Пусть размерность векторных представлений равна D. Возьмём линейные преобразования вектора h_i: q_i, k_i и v_i. Будем называть их запросом, ключом и значением соответственно:

$$q_i = W_Q\, h_i; \quad k_i = W_K\, h_i; \quad v_i = W_V\, h_i,$$

где матрицы W_Q, W_K, W_V — размера $P \times D$, где число P — какая-то константа.

Тогда результатом *головы внимания* назовём следующую величину:

$$\text{head}_i = \sum_j \alpha_{ij}\, v_j,$$

где j принимает все значения от 1 до длины предложения, а α_{ij} — это веса внимания, вычисляемые по следующим формулам:

$$\alpha_{ij} = \frac{\exp(k_j^T\, q_i)}{\sum_l \exp(k_l^T\, q_i)}.$$

По смыслу это означает следующее. Матрица W_Q при умножении на вектор даёт нам запрос на поиск информации в других словах, а матрица W_K даёт ключ поиска, по которому информация находится в других словах. Вектора h_i, h_j, под воздействием этих матриц

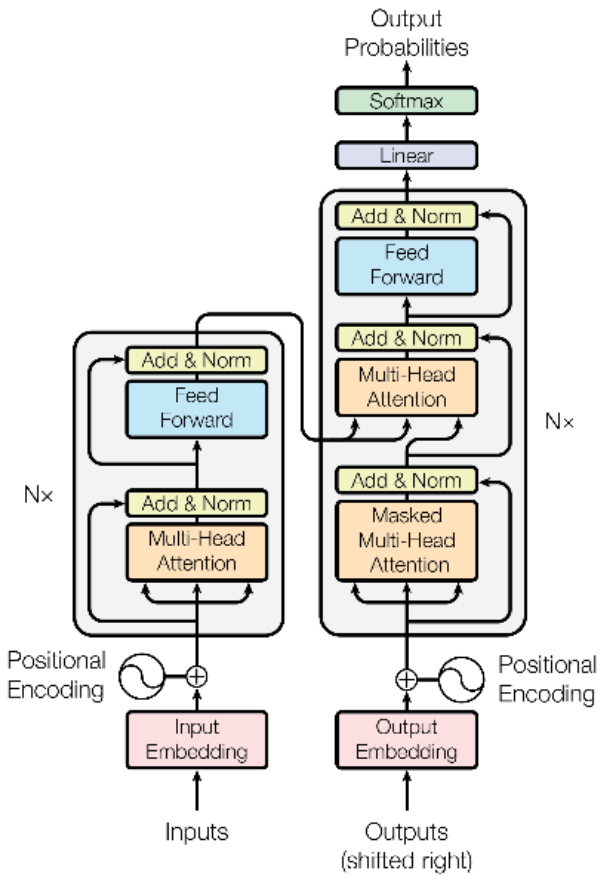

Рис. 1: Общая архитектура кодировщика нейросети Transformer.

дают веса внимания. Далее вектора всех слов предложения под действием матрицы W_V, которая при умножении на вектор достаёт из него нужную информацию, суммируются в соответствии с найденными весами. Подробно данная процедура представлена на Рис. 2 слева.

На каждом слое внимания, к вектору h_i независимо применяются несколько голов внимания, а результаты их работы конкатенируются. Чтобы сохранить размерность пространства после слоя внимания, размерность головы внимания P выбирается так, чтобы после конкатенации всех голов размерность была снова равна D. Подробно данная процедура представлена на Рис. 2 справа.

Обычно в архитектуре используется несколько голов, так как это позволяет модели одновременно выучить различные типы агрегирования информации. Также это позволяет сократить количество параметров на слое.

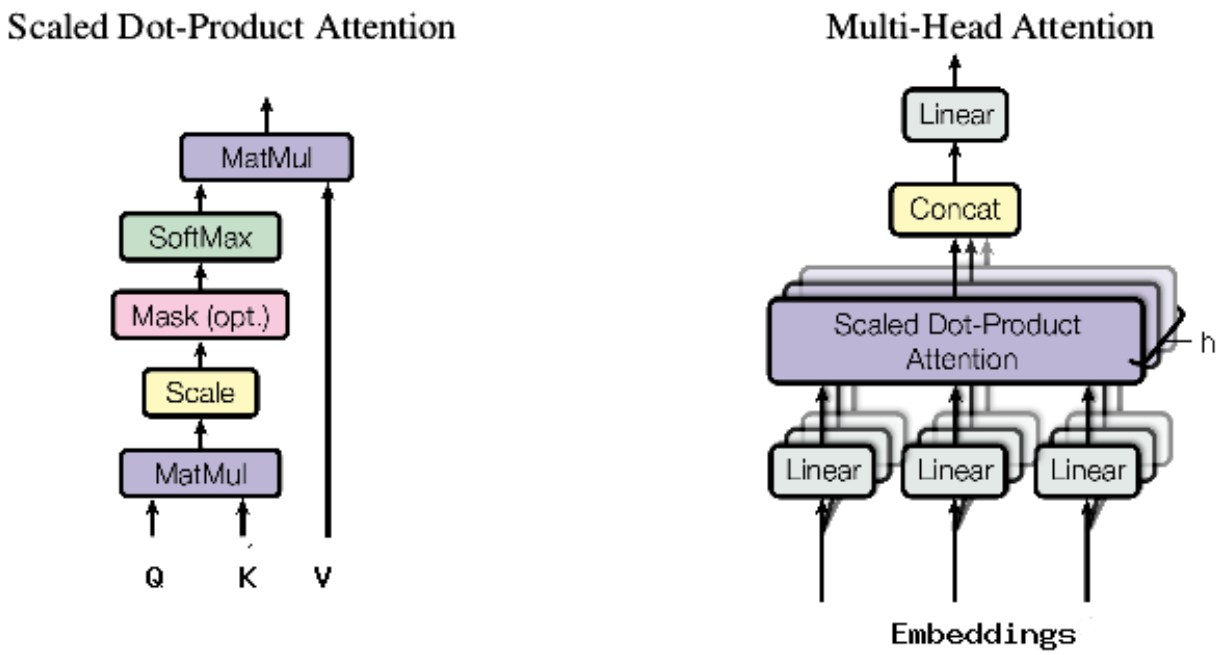

Рис. 2: Механизм внимания в модели Transformer.

По смыслу данная архитектура позволяет на каждом слое контексту влиять на каждое векторное представление каждого слова, обогащая его смыслом. На выходе каждого блока Transformer, как и сети в целом, выдается по одному контекстному вектору на каждое слово входа.

Декодировщик отличается от кодировщика двумя вещами. Во-первых, в механизме внимания подаются токены только предшествующие данному. Это реализуется с помощью механизма маскирования. Во-вторых, в каждом блоке после слоя внимания добавляется слой внимания на выходы энкодера. В таком внимании в качестве ключей используются токены выхода, а в качестве запросов — представления токенов входа. Данный механизм позволяет в каждом блоке использовать информацию о входе, что необходимо в задаче перевода.

Обучаются современные нейросетевые архитектуры методом Adam [7] основанным на процедуре стохастического градиентного спуска. Суть этих методов заключается в том, чтобы вычислить градиент функции потерь по параметрам нейросети и изменить эти параметры в соответствии с направлением вычисленного антиградиента для того, чтобы

уменьшить значение функции потерь.

В данной работе, как и в предшествующей для обучения использовалась именно архитектура Transformer с описанными методами оптимизации. Более подробно с деталями проведения экспериментов можно будет ознакомиться в секции экспериментов.

4.2 Оценка качества перевода

Оценка качества в задаче перевода достаточно сложная проблема. Пусть у нас есть какое-то количество контрольных переводов, полученных с помощью качественной человеческой разметки. Тогда с точки зрения машинного обучения, на каждой итерации при генерации перевода, мы решаем задачу классификации на множество классов, выбираем какое слово выбрать. Тогда следующее слово в разметке является правильным классом и мы можем использовать стандартные метрики классификации, такие как точность и полнота.

Теперь стоит заметить, что генерация следующего слова зависит от всего префикса, то есть корректно оценивать такие метрики только подставляя префикс этого самого эталонного перевода. Но если наша модель, например, решила изменить порядок слов при переводе, то для полученного префикса с другим порядком слов уже невозможно посчитать метрики пословной классификации.

Для борьбы с этой проблемой была придумана метрика BLEU [8], имеющая достаточно хорошую корреляцию с оценками людей. Метрика считает точность для n-грамм различной длины из перевода в эталонном переводе. После подсчёта метрики для различных длин n-грамм, берётся геометрическое среднее между ними. Для n-грамм длины 1, метрика для перевода t и эталона x считается по следующей формуле:

$$P_1 = \frac{\sum_{w \in t}[w \in x]}{|x|}$$

Авторами метрики было показано, что BLEU имеет высокую корреляцию с качеством перевода при сравнении систем и человеческих экспертов друг с другом. Для сравнения систем, авторы использовали системы машинного перевода различной сложности, а для сравнения людей, использовали экспертов с различными уровнями владения целевого языка.

Метрика рассчитывается на тестовых корпусах, созданных профессиональными переводчиками. В процессе перевода, переводчикам запрещалось пользоваться какими-либо системами автоматического перевода. Для англо-финского направления брались тестовые корпуса с конференции WMT-2017 [9].

4.3 Оценка качества циклического перевода

В данной работе дополнительной целью является увеличение качества консистентности перевода. В конкретном случае, с учётом постановки задачи, требуется оценивать то, насколько сильно циклический перевод текста совпадает с самим исходным текстом. В предыдущей секции была описана метрика BLEU, которая позволяет мерить схожесть машинного перевода с эталонным. В данной работе предлагается использовать эту же метрику для измерения схожести циклического перевода с оригинальным текстом. Далее в экспериментах будем обозначать данную метрику CycleBLEU.

Ожидается, что даже если от использования процедуры совместного обучения роста качества прямой модели в BLEU не будет, то рост CycleBLEU сам по себе является хорошим признаком. Это можно мотивировать тем, что при использовании такой системы несовпадения циклических переводов с оригинальным текстом будут реже вводить пользователя в недоумение.

5 Эксперименты

В данном разделе будут предоставлены условия проведения экспериментов, а также основные результаты по обучению модели перевода с учётом циклических переводов. Сначала описываются результаты при обучения с нуля, также как и в оригинальной работе [2]. Однако в силу ограниченности вычислительных ресурсов, языковые модели использоваться не будут. Будет проверена состоятельность такого облегчённого по сравнению с оригиналом метода обучения.

Далее будет проведён анализ использования подхода при дообучении уже предобученной модели перевода с помощью циклических переводов. При этом метод будет сравниваться с самой предобученной моделью, а также с моделью, доучивавшейся без циклических переводов на обычную авторегрессионную функцию потерь.

5.1 Детали экспериментов

Для всех экспериментов использовались модели, основанные на описанной архитектуре Transformer.

Для экспериментов при обучении с нуля использовалась конфигурация модели tiny. В ней промежуточная размерность составляет 256, размерность внутри полносвязанных слоёв — 1024, используется 4 головы внимания на каждом слое, а всего слоёв по 6 в кодировщике и декодировщике. Данная конфигурация была выбрана для более быстрого обучения и проверки сходимости модели, использовавшей циклические переводы. Обычно, несостоятельные подходы можно выявить уже при малом количестве параметров в архитектуре.

Также во всех экспериментных моделях использовался механизм относительного позиционного кодирования [10]. При его использовании в механизме внимания при подсчете произведения ключей и запросов добавляется информация, зависящая от расстояния между токенами. Это усложнение было перенесено из предобученных моделей, которые использовались в экспериментах.

Для экспериментов с доучиванием, использовались предобученные модели конфигурации base. В каждой из них по 6 слоёв в кодировщике и декодировщике, со размерностью

внутренних представлений — 512, и 2048 — внутри полносвязанных слоёв, количество голов внимания — 12.

Оптимизировались данные модели методом Adam, с нагревом в течение 10000 итераций. После нагрева коэффициент шага уменьшался. При этом максимальное значение коэффициента за время обучения составляло $4 \cdot 10^{-3}$. При доучивании моделей, нагрев происходил в течении 2000 итераций, максимальное значение коэффициента — 10^{-5}. Батчи формировались размером в среднем по 500 текстов.

В качестве целевых языков для улучшения были выбраны англо-финское и русско-казахское направления. Данные для направлений собирались методом обхода мультиязычных сайтов, в адресах которых различия заключаются исключительно в тэге языка. Тексты с данных сайтов выравнивались по предложениям и добавлялись в обучающую выборку. Объем таких данных для каждого из направлений составляет порядка 50 млн. пар предложений.

Перед подачей на вход модели Transformer, тексты обрабатывались с помощью процедуры BPE [6]. В этом подходе слова делятся наиболее частотные подслова, существенно сокращая суммарный размер входного словаря. Общий размер словаря для прямый и обратных моделей в обоих направлениях составляет 32000 токенов.

Также к описанным данным добавлялись синтетические данные, полученные путем перевода одноязычных текстов целевого языка с помощью обратной модели. Данный способ может уменьшить прирост от использования циклических переводов, поэтому для экспериментов были взяты направления с малым количеством одноязыковых документов для целевого языка. Объем одноязыковых данных для генерации синтетических данных для русско-казахского составляет 5 млн. предложений, что в 10 раз меньше чем количество параллельных данных. Для финского, число одноязыкового корпуса составляет 50 млн. предложений, то есть сопоставимо с числом параллельных данных.

Обычно используются одноязыковые корпуса большего объема, чем параллельные корпуса. Однако от этих корпусов требуется высокое качество, иначе синтетика ухудшит итоговое качество перевода. Для финского и казахского языков не удалось найти такое количество качественных данных.

Во всех экспериментах с циклической функцией потерь проводился подбор значения

её веса относительно авторегрессионной функции потерь, которая также использовалась. После подбора параметра, оптимальное значение составило 10^{-3}.

Для обучения и валидации модели применялся сервер с 8 GPU Tesla M40. При этом при обработке батча, батч равномерно делился между устройствами а перед обновлением модели, результаты агрегировались по всем устройствам.

5.2 Эксперименты с обучением с нуля

В оригинальной статье [2] предполагалось одновременное обучение прямой и обратной моделей, а также взвешивание с помощью языковых моделей. Для хранения всех этих моделей и их градиентов в памяти размер батча на используемых видеокартах придётся значительно уменьшить. Чтобы не сходимость из-за уменьшения батча не ухудшилась, можно аккумулировать градиенты, делая обновление параметров не на каждом батче данных. Однако такое решение значительно увеличит время обучения, которое и так достаточно долгое. Обучение обычной прямой модели занимает порядка двух недель.

Используя оригинальную идею авторов, воспроизведем ее частично исходя из возможных ресурсов. Поэтому исключим использование языковых моделей для взвешивания и будем оптимизировать только параметры прямой идеи с помощью градиентов от циклических переводов. Также не будем совместно обучать прямую и обратную модель, так как это увеличит количество обучаемых параметров вдвое. Вместо этого будем учить прямую модель с использованием предобученной обратной модели. При такой конфигурации параметры обратной модели всё равно придётся хранить в памяти вычислительного устройства, но не нужно хранить параметры оптимизатора.

Описанная конфигурация сильно отличается от той, что использовалась в оригинальной работе [2]. К сожалению, использовавшиеся там рекуррентные модели перевода устарели, а в современных архитектурах Transformer существенно больше параметров. Таким образом, в проводимых экспериментах исследуется применимость идей из оригинальной работы в современных моделях.

Для обучения с нуля начнём с моделей с небольшим количеством параметров, то есть конфигурацию tiny. На ней проверим работоспособность идеи. Обучение проводится на направлении en-fi на описанных данных.

Результаты обучения спустя 30000 итераций представлены в таблице 1. Как можно заметить, качество при использовании циклических переводов заметно ухудшилось. Проседание качества можно уменьшить, устремляя вес функции потерь циклических переводов к нулю, однако прироста качества этот метод не дает. Более того, нет роста качества и CycleBLEU, хотя циклические переводы растят их качество практически напрямую.

Модель	BLEU на вал.	CycleBLEU на вал.
Базовая	12.0	50.0
Базовая+обратная	10.0	42.0

Таблица 1: Результаты при обучении с нуля

Этот эффект можно объяснить тем, что в начале обучения качество переводов прямой модели крайне низкое и не соответствует реальным переводам текста. Соответственно для этого плохого перевода обратный перевод также не является хорошим переводом с точки зрения обратной модели. Соответсвенно сигнал от циклической функции потерь крайне шумный и несёт мало полезного вклада. Именно этот факт подводит нас к идее дообучения предобученных прямых моделей с использованием процедуры циклических переводов.

5.3 Эксперименты с доучиванием моделей

Переходим к экспериментам с предобученными моделями.

Для процедуры дообучения были взяты модели конфигурации base Transformer. Модели были обучены до сходимости по метрике BLEU на тестовых корпусах с помощью авторегрессионной функции потерь (1).

Процедура дообучения проводилась на тех же данных, что и предобучение, однако коэффициент шага был существенно меньше. Из-за этого качество модели на дообучении может расти и без использования процедуры циклического перевода (2). Поэтому сравнивать качество предложенного метода будем не только с предобученной моделью, но и с дообученной только с помощью авторегрессионной функции потерь (1).

Результаты доучивания для англо-финского можно увидеть в таблице 2. Как можно заметить, удалось вырастить качество на 0.5 BLEU, что достаточно неплохой результат.

Таким образом, можно утверждать, что доучивание с циклической функцией потерь (3) улучшает качество перевода. Также стоит отметить, что рост качества достаточно большой при любом виде доучивания, так что, как и предполагалось, доучивание с авторегрессионной функцией потерь (1) также заметно улучшает качество.

Модель	BLEU на вал.	CycleBLEU на вал.
Без доучивания	23.4	47.0
Базовая	25.0	55.0
Базовая+обратная	25.5	54.5

Таблица 2: Результаты дообучения с циклической функцией потерь для англо-финского направления

Также, если сравнивать результаты для CycleBLEU, можно увидеть, что результаты практически совпали для обучения с циклической функцией потерь и без, хотя циклическая функция потерь (3) должна улучшать качество циклических переводов. Этот неожиданный результат можно объяснить двумя причинами: особенностью данного тестового корпуса, либо наличием заметного числа синтетических обратных переводах в обучающем корпусе прямой модели.

Перейдём теперь к результатам для русско-казахского направления. Сравнение можно увидеть в таб. 3. Как можно заметить, рост BLEU составляет порядка 0.25. Это не очень большое значение. Также прирост относительно предобученной модели меньше по сравнению с en-fi и составляет 0.5 BLEU.

Модель	BLEU на вал.	CycleBLEU на вал.
Без доучивания	19.5	41.5
Базовая	19.80	42.5
Базовая+обратная	20.05	44.9

Таблица 3: Результаты дообучения с циклической функцией потерь для русско-казахского направления

При этом, можно заметить, что CycleBLEU на русско-казахском растёт на 2.4 пункта.

Совмещая эти приросты качества, можно считать, что процедура циклических переводов улучшает качество и на этом направлении. Больший по сравнению с англо-финским прирост CycleBLEU можно объяснить существенно меньшей долей синтетических обратных переводов в обучении.

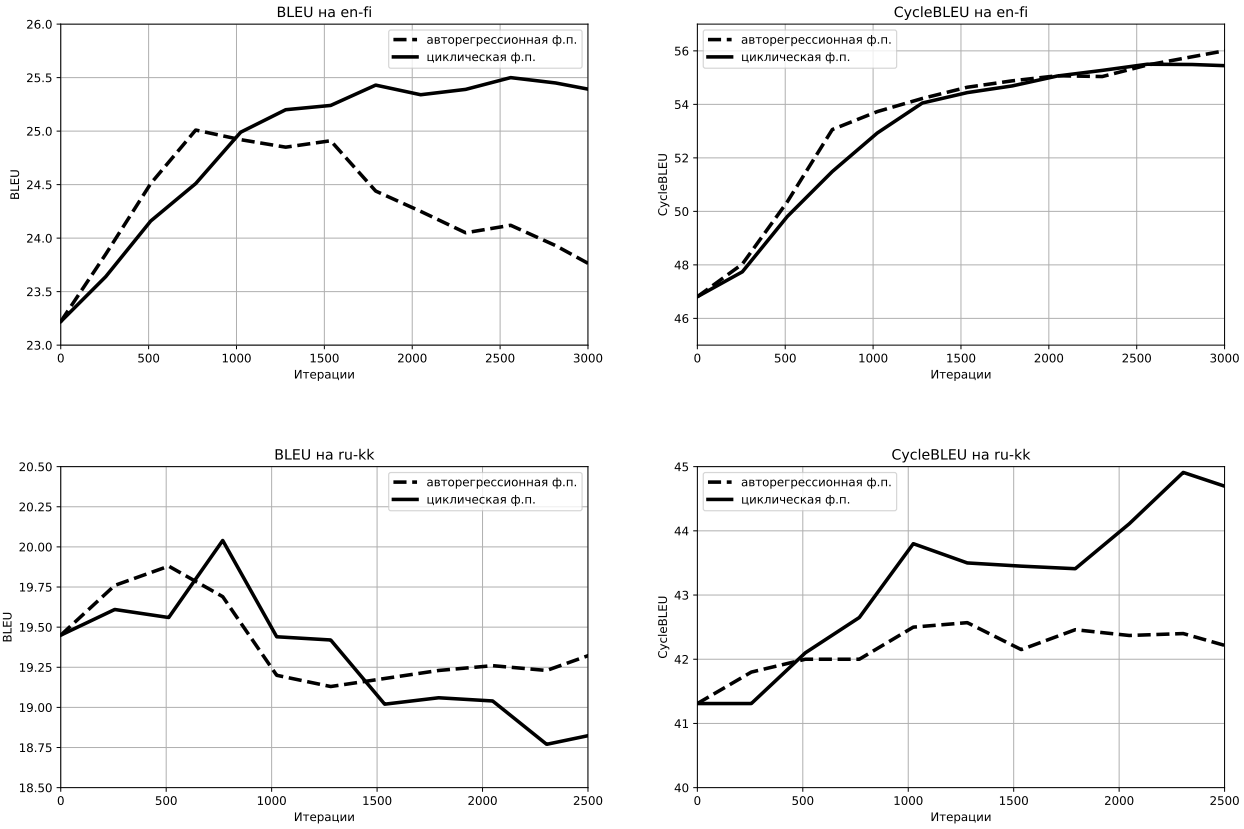

Рис. 3: Графики дообучения моделей с использованием циклической функции потерь и без. Верхняя строка соответствует направлению en-fi, нижняя — направление ru-kk. Первая колонка соответствует графикам BLEU, вторая — графикам CycleBLEU.

На графике 3 можно увидеть BLEU и CycleBLEU во время обучения для англо-финском и русско-казахском направлениях. При сравнении модели рассматривались на итерации, на которой они показывали самое лучшее качество. Как можно увидеть, на англо-финском рост качества модели с циклической функцией потерь больше и, в целом, более стабильный. При этом, относительного роста на CycleBLEU не наблюдается, что, как и говорилось, связано с наличием большой доли синтетических данных в обучении. На русско-казахском, наоборот, почти не наблюдается роста BLEU, зато виден прирост стабильный прирост

CycleBLEU.

В итоге, общее число итераций, в течение которых проводилось дообучение, не превосходило 3000, что в 10-100 раз меньше, чем занимает обучение такой архитектуры с нуля. Таким образом, прироста от использования циклических переводов удаётся добиться быстрее чем методом, описанным в оригинальной работе [2].

В целом, можно отметить, что удалось добиться общего прироста качества на обоих направлениях, на которых проводились эксперименты. Это можно объяснить качеством обратной модели, которая использовалась для улучшения качества, а для целевого языка обратной модели данных было больше чем для прямой. В итоге прирост получился меньше, чем в оригинальной работе, но процедура улучшения получилась намного более простой с вычислительной точки зрения и быстрой. При этом, используя некоторые из описанных в [2] идей, удалось применить подход с использованием циклическим переводов в современных моделях.

6 Заключение

В данной работе был проанализирован подход к улучшению качества машинного перевода методом использования обратных моделей и циклических переводов, полученных с их помощью. Удалось показать, что использование данного подхода для современных архитектур моделей машинного перевода как увеличивает качество прямых моделей на тестовых наборах, так и увеличивает консистентность переводов с обратной моделью. То есть в среднем, циклические переводы, которые могут быть получены для самопроверки, больше совпадают с оригинальными текстами. Это означает большую стабильность и увеличивает доверие к точности передаваемого при переводе смысла.

Также при разработке метода использования циклических переводов были рассмотрены теоретические основания и вероятностные модели, к нему приводящие. В итоге было получено полное описание функции потерь, являющейся нижней оценкой логарифма правдоподобия вероятностной модели, а также способ получения из неё оценки градиента по параметрам модели. Данное описание модели может быть использовано для улучшения и модификации подхода на различных этапах. Формульные значения градиентов при этом совпали с полученными в предшествующей работе [2], но там не было представлено полных обоснований для используемых формул, и некоторые из них были получены эвристически.

В рамках адаптации предложенного подхода к современным архитектурам моделей, в данной работе реализована идея использования предобученных моделей перевода в процедуре обучения с циклическими переводами. Данная идея существенно повысила стабильность подхода и позволила сократить время обучения и количество требуемых вычислительных ресурсов.

Данная работа раскрывает потенциал использования обратных переводов и позволяет в будущем перейти к использованию языковых моделей в процессе дообучения по аналогии с предшествующей работой. Также в будущем возможна модификация вероятностной модели циклических переводов, являющейся основой данного подхода, например, с помощью регуляризации.

Список литературы

[1] *Felix Stahlberg* (2019) Neural Machine Translation: A Review

[2] *Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, Wei-Ying Ma* Dual Learning for Machine Translation, 30th Conference on Neural Information Processing Systems (NIPS 2016).

[3] *Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio* (2015) Neural Machine Translation by Jointly Learning to Align and Translate. ICLR

[4] *P. Brown; John Cocke, S. Della Pietra, V. Della Pietra, Frederick Jelinek, Robert L. Mercer, P. Roossin* A statistical approach to language translation. (1988) Coling'88. Association for Computational Linguistics. 1: 71–76. Retrieved 22 March 2015.

[5] *Tillmann, Christoph, Xia, Fei* (2003) A Phrase-based Unigram Model for Statistical Machine Translation, Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers, pp. 106-108

[6] *Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.* (2017). Attention Is All You Need. NIPS.

[7] *Kingma, Diederik & Ba, Jimmy* (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[8] *Papineni, Kishore Roukos, Salim Ward, Todd Zhu, Wei-Jing* (2002) Bleu: a Method for Automatic Evaluation of Machine Translation, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311-318

[9] *Bojar, Ondřej and Chatterjee, Rajen and Federmann, et al.* Findings of the 2017 Conference on Machine Translation (WMT17). 2017. Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers

[10] *Peter Shaw, Jakob Uszkoreit, Ashish Vaswani* (2017) Self-Attention with Relative Position Representations. NAACL