МЕСТО ПУБЛИКАЦИИ:
Митренина О. В. Машинный перевод // Прикладная и компьютерная лингвистика / Николаев И.С., Митренина О.В., Ландо Т.М. (ред.), М.: URSS, 2017. С. 156–189.

Митренина О. В.

Глава 7. Машинный перевод

1. Три подхода к машинному переводу

Машинный перевод был первой задачей компьютерной лингвистики. Точнее, второй. Первой была дешифровка, а идея машинного перевода возникла из дешифровки естественным образом.

Первый компьютер появился в 1938 году — его собрал молодой немецкий инженер Конрад Цузе, сдвинув столы в гостиной родительского дома. В 1939 году началась Вторая мировая война, и вскоре компьютеры начали использоваться для расшифровки перехваченных сообщений противника. Они позволили сократить время дешифровки с нескольких недель до нескольких часов.

После войны американский математик Уоррен Уивер предложил рассмотреть задачу автоматического перевода именно как дешифровку. «Когда я вижу текст на русском языке, я говорю себе, что на самом деле он написан по-английски и зашифрован при помощи странных знаков. И мне надо его просто расшифровать», — так описал Уивер задачу машинного перевода в своем письме математику и философу Норберту Винеру. Дату этого письма — 4 марта 1947 года — можно считать днем рождения машинного перевода как научного направления.

Но идея перевода как дешифровки была понятна далеко не всем. Ведь люди переводят совсем не так, как работает дешифровщик. Человек должен знать грамматику обоих языков и уметь пользоваться словарем. Тогда можно понять, что в предложении *Маша ела кашу* деятелем является Маша, а ее действием — поедание какого-то продукта, которым оказывается каша. Осознав смысл предложения и роли входящих в него слов, человек строит новое предложение на другом языке.

Дешифровщик действует иначе. Зашифрованный текст — это не случайная цепочка знаков, в нем должны быть какие-то закономерности. Дешифровщик отмечает частоту отдельных элементов текста и частоту взаимной встречаемости элементов. Он может запомнить, в каком окружении появляются те или иные куски непонятного текста. Эти и другие «отклонения от случайности» позволяют ему найти ключ к преобразованию закодированного сообщения из непонятной цепочки символов в текст на понятном языке. Так видел задачу машинного перевода Уоррен Уивер, который и сам во время войны работал шифровальщиком.

Но история распорядилась иначе, и методы дешифровки понадобились разработчикам на 50 лет позже, чем предполагал Уивер. А до этого машину учили переводить тем же способом, которым действуют переводчики-люди — с помощью словарей и грамматик.

Сегодня существуют три разных подхода к машинному переводу:

1) Перевод на основе правил (rule-based machine translation, RBMT) действует по той же схеме, что и человек-переводчик. Для нее необходим как можно более подробный словарь и очень подробная грамматика обоих языков. Все ранние системы переводов были основаны на правилах.

2) Статистический машинный перевод (statistical machine translation, SMT) обходится без словаря и грамматики, он работает только на основе методов машинного обучения. Компьютеру потребуется очень много пар «предложение + его перевод». Он запомнит, как переводились двойки или тройки слов, в каком окружении они встречались, где в предложении находились. После этого, получив для обработки новое предложение, он сможет выбрать его наиболее вероятный перевод. При этом он будет учитывать только окружение, в котором встречалось то или иное слово, и частоту различных цепочек слов.

3) Гибридный перевод (hybrid machine translation, HMT) — это комбинация правил и статистики. Совмещать два этих подхода можно разными способами, поэтому гибридные переводчики могут быть разных типов. Сейчас это самый современный подход, почти все разработчики стремятся использовать гибридные технологии.

При машинном переводе текст обрабатывается компьютером без участия человека. А если текст переводится человеком с использованием различных компьютерных технологий, то говорят об **автоматизированном переводе** (CAT, computer-aided translation или computer-assisted translation), но это не машинный перевод, и мы его рассматривать не будем.

2. Перевод на основе правил

2.1. Три способа перевода с помощью правил

В 1954 году американцы продемонстрировали миру первую действующую программу машинного перевода. Ее разработали совместно фирма IBM и Джорджтаунский университет. В честь университета презентацию программы назвали Джорджтаунским экспериментом. На глазах у зрителей компьютер перевел 49 заранее отобранных предложений с русского на английский язык. Он использовала словарь из 250 слов и грамматику, состоящую из шести синтаксических правил.

Долгое время все системы машинного перевода разрабатывали именно этот подход — в машину загружались все более сложные словари и правила анализа слов и предложений.

Принято выделять три разных типа перевода на основе правил:

- трансферные системы (исходное предложение → препарированное исходное предложение → предложение на другом языке);
- системы пословного перевода (слова исходного текста → слова перевода);
- интерлингвистичесие системы (исходный текст → описание его смысла на универсальном языке-посреднике → текст перевода).

Трансферная система (transfer-based machine translation, transfer systems) в наибольшей степени подражает человеку-переводчику. Она анализирует исходное предложение, переводит его слова, выясняет их роли, а затем с помощью грамматики собирает новое предложение на конечном языке.

Система пословного перевода (word-by-word machine translation, другие названия — direct systems, dictionary based machine translation) — это самый примитивный подход. В предложении переводятся все слова, а текст правится минимально. При этом получается перевод хотя и не очень гладкий, но в большинстве случаев понятный. Этот подход использовался в ранних программах, он и сейчас иногда применяется, например, для упрощения работы переводчиков-людей.

Интерлингвистичесие системы (interlingual systems) используют язык-посредник, который называется **интерлингвой** (interlingua). Строго говоря, это даже не язык в привычном для нас смысле, а некоторое формальное представление смысла человеческой речи и мыслей. На заре компьютерной лингвистики идея создания такой системы была очень популярна [Беляева, Откупщикова 1996], но создать полноценный универсальный вспомогательный язык ученым пока не удалось.

Пословные переводчики и интерлингвистические системы сейчас практически не используются. Большинство RBMT систем применяют трансферный подход, поэтому мы рассмотрим его чуть подробнее.

2.2. Трансферный подход

Практически все компьютерные системы переводят текст по предложениям, поэтому текст должен быть разбит на предложения. В каждом из них надо выделить входящие в него слова, знаки препинания и другие элементы. Полученные элементы называются **токенами** (tokens), а сам процесс — **токенизацией** (tokenization).

Далее **токены** (то есть, слова и знаки) необходимо проанализировать, чтобы понять их роль в предложении. Об этом рассказывается во второй главе данной книги, которая посвящена компьютерной морфологии. В результате для каждого слова будут определена его часть речи и его грамматические значения. Кроме того, для целей перевода необходимо сопоставить каждое слово с нужным разделом компьютерного словаря. Об этих словарях мы поговорим подробнее чуть позже.

После того, как все токены проанализированы, необходимо понять структуру предложения — найти связи между словами, объединить слова в группы. Об этом рассказывается во второй главе нашей книги, посвященной компьютерному синтаксису. Так, многие переводчики на раннем этапе собирают именные группы. Например, сочетание *Добрые животные* будет отмечено как целостная именная группа.

Когда структура предложения будет ясна, компьютер строит новое предложение на конечном языке, используя переводы слов. Но это в идеале. На практике всегда оказывается, что существующих правил обработки недостаточно, а описание слов в словаре должно быть более подробным. После этого начинается этап совершенствования системы. Разработчик анализирует переводы и пополняет словари и правила новыми условиями, чтобы компьютер мог учитывать смысловые и формальные связи. Например, английское слово *plant* имеет два значения: *растение* и *завод*, поэтому сочетание *the strike at the plant* может быть ошибочно переведено как *забастовка на растении.* Но предлог *at* в сочетании *at the plant* употребляется обычно тогда, когда речь идет о заводе. Поэтому можно добавить в систему правило: если слово *plant* сочетается с предлогом *at*, значит, переводить его надо как *завод*.

Правильный вариант перевода можно выбирать на основе семантического типа слов. Например, если английский глагол *to press* употребляется с элементом одежды (*the dress* — «платье»), то его надо переводить как «гладить» или «погладить», а не «нажимать».

Улучшить качество текста помогает **база памяти переводов** (translation memory) — сохраненные заранее сегменты текста и их переводы, выполненные человеком. Это могут сочетания слов и целые предложения, например, идиомы или фрагменты деловой и личной переписки. При переводе они могут вставляться в текст автоматически или по указанию пользователя.

Способы лингвистической настройки перевода весьма разнообразны, но они требуют много часов (точнее, много лет) работы квалифицированных лингвистов. Наверное, самые подробные словари для русского языка существуют в компании PROMT. Они разрабатываются уже более 20 лет усилиями многих лингвистов и инженеров и могут быть интегрированы не только в системы перевода, но и в другие системы обработки языка.

2.3. Пример словарей и грамматик компании PROMT

Рассмотрим систему словарей компании PROMT — ведущей российской организации, занимающейся машинным переводом. Сейчас переводчики PROMT используют гибридные технологии, но их основу составляет мощная система словарей и грамматик,

позволяющая компании выигрывать международные конкурсы по машинному переводу на русский язык.

Система словарей в PROMTe содержит:

1. словари основ;
2. таблицы флексий;
3. вспомогательные таблицы.

В **словаре основ** хранится (1) базовая форма каждого слова, (2) его псевдооснова, (3) его грамматические и семантические признаки, а также (4) его переводы на другой язык с подсказками, какой перевод в каких случаях использовать.

Как, например, устроена словарная статья для слова *площадь*? Она хранит его базовую форму *площадь*, чтобы было понятно, о каком слове идет речь. Кроме того, в словарной статье хранится псевдооснова *площад* — именно эта часть слова не меняется при изменении числа и падежа. Некоторые грамматические признаки этого слова записываются в словарную статью, а другие признаки хранятся в виде ссылок. В словарную статью можно записать, что это существительное мужского рода. И можно дать ссылку на ту строчку в таблице флексий, где хранятся окончания этого слова для разных падежей. Это будет та же строчка, что и у слова *кровать*, потому два этих слова изменяются одинаково.

Семантические признаки удобно привязывать к переводу. Английский слово *площадь* можно перевести тремя способами: *area* (если речь идет о территории), *square* (если подразумевается городская площадь) или *ploschad* (при транслитерации названий). Контекст слова *площадь* может подсказывать, как правильно его перевести. Например, если за ним идут цифры, а затем единицы измерения (*100 кв.м.*), то тогда это сочетание переводится как *square of*. Вся эта информация хранится в словаре основ.

В **таблице флексий** хранятся все типы изменения слов.

Вспомогательные таблицы созданы для обработки нестандартных случаев слов. Таких таблиц несколько. Это базы префиксов и постфиксов, то есть, начальных и конечных частей слов, а также база имен и географических названий.

В базе префиксов указаны, например, переводы начальных частей слов:

восточно — East
восьмидесяти — eighty-
высоко — high-
вэб — web

В базе постфиксов хранятся не только окончания слов, но и их предположительные части речи, а также похожее слово с таким же окончанием. Вот примеры записей из таблицы постфиксов:

*ойл — NOUN — Казахойл
*инвест — NOUN — Связьинвест
*дума — NOUN — Мосгордума
*банк — NOUN — Райффайзенбанк

В базе имен и географических названий указаны переводы многих личных имен с указанием их рода, типа склонения и некоторых других полезных сведений.

В процессе лингвистической настройки словари активно пополняются, в них заносятся новые значения и условия. Но, к сожалению, эти кропотливо собранные данные можно использовать только для одной языковой пары (например, для перевода с английского на русский) и, как правило, только для одной системы.

В этом отношении данные для статистического перевода гораздо более универсальны.

3. Статистический машинный перевод

3.1. Главная формула перевода

В основе статистического перевода лежат разработки американского математика Клода Шеннона, которые он вел в совершенно другой области. Во время Второй мировой войны немцы научились расшифровывать телефонные переговоры между правительствами США и Великобритании. Тогда решено было разработать новую систему передачи голоса по телефону. Предполагалось сжимать речевой сигнал и маскировать его с помощью шума. Клод Шеннон придумал математический аппарат, который позволял дешифровать полученный на выходе звуковой поток и выделять из шума исходный сигнал.

Разработки Шеннона позже пригодились и для машинного перевода. Ведь можно представить, что имеющееся у нас исходной предложение мы получили в результате маскировки сигнала с помощью шума. А его перевод — это на самом деле исходное переданное нам предложение, которое нужно декодировать.

Если у нас есть предложение *x* в исходном языке, то задача машинного сводится к поиску в конечном языке такого предложения *y*, которое с наибольшей вероятностью является переводом предложения *x*. Иными словами, нас интересует условная вероятность — вероятность предложения *y* при наличии предложения x. Она записывается как *p(y|x)*.

Например, мы переводим английское предложение *The dog is hungry.* Теоретически, его переводом может оказаться любое русское предложение *y*:

y_1 = *Собака сидит на крыше.*
y_2 =*Собака голодная.*
y_3 =*Собака голодный.*
y_4 = *Собака есть голодная.*
y_5 = *Голодная голодная собака.*
…

У каждого из этих предложений есть своя вероятность быть переводом предложения *x*. Можно записать эти вероятности как $p(y_1|x)$, $p(y_2|x)$, $p(y_3|x)$ и т.д.

Из всех этих предложений y_n компьютер должен выбрать самое вероятное, при условии, что у нас есть предложение *x*. Или, если записать это с помощью формул, мы ищем вот что:

$$(1)\ \arg\max_y p(y|x)$$

Но эту формулу можно переписать с помощью теоремы Байеса, о которой говорилось в пятой главе, посвященной методам машинного обучения.

$$(2)\ \arg\max_y p(y|x) = \arg\max_y \frac{p(y)\times p(x|y)}{p(x)}$$

Получившуюся формулу можно упростить. Значение в знаменателе *p(x)* — это вероятность появления исходного предложения *x*. Она является постоянной величиной и не влияет на поиск такого *y*, при котором вероятностная формула принимает максимальное значение. Поэтому знаменатель *p(x)* можно убрать:

$$(3)\ \arg\max_y p(y|x) = \arg\max_y p(y) \times p(x|y)$$

Иными словами, нам необходимо найти, при каком *y* получается максимальное произведение двух величин. Первая из них — это вероятность *p(y)*, вероятность появления

предложения *y* в языке. Понятно, что такая вероятность будет выше у предложения *Собака голодная,* чем у предложения *Собака голодный.* Эта вероятность вычисляется с помощью **модели языка**, которой будет посвящен следующий раздел.

Множитель $p(x|y)$ в формуле (3) — это «обратный перевод», вероятность того, что конечное предложение *y* можно перевести с помощью исходного предложения *x*. Она вычисляется с помощью **модели перевода**, о которой мы тоже поговорим в этой главе. В нашем примере это вероятность того, что предложение *Собака голодный* можно перевести на английский как *The dog is hungry.*

Но тут может возникнуть вопрос. Неужели формула Байеса упрощает ситуацию? Ведь фактически, нам надо считать то же самое, вероятность перевода с одного языка на другой, но только в другую сторону. Но здесь нужно вспомнить, что теоретически в качестве кандидата у нас может появиться абсолютно любое предложение, составленное из слов конечного языка. Кто знает, что нам предложит машина! Параметр $p(y)$ позволяет нам оценить, насколько естественно звучит получившееся предложение.

3.2. Модель языка и цепи Маркова

Модель языка определяет вероятности появления того или иного предложения. Ведь в русском языке можно сказать не только *Собака виляет хвостом,* но и *Хвост виляет собакой,* а также *Хвосты виляет хвосты* или, допустим, *Хвост хвост хвост*. Никто из нас не может гарантировать, что ему никогда в жизни не встретится предложение *Хвост хвост хвост*.

Но вероятность появления этих предложений в языке разная. Очевидно, что *Собака виляет хвостом* — это более вероятное фраза, чем *Собака виляет шлейфом*. А откуда мы это знаем? Мы просто помним, как часто нам попадались в жизни эти предложения или их части. Вряд ли кому-то до прочтения этой книги попадалось предложение *Собака виляет шлейфом*.

Компьютер тоже может запомнить, насколько вероятно появление того или иного предложения в языке. Для этого ему надо «набраться языкового опыта» — посмотреть, какие предложения и словосочетания встречаются в языке. Чем больше текстов он «посмотрит», тем лучше сможет предсказать вероятность появления какого-либо предложения. Причем, в идеале эта вероятность никогда не должна равняться нулю. Разве можно быть уверенным, что какое-то предложение никогда и нигде не встретится?

Модель языка или **языковая модель** (language model) — это способ вычислять вероятность для всех теоретически возможных предложений языка. Эти способы могут быть разными, поэтому и модели языка существуют разные.

Чтобы построить модель языка, машине нужен корпус — большой набор текстов. Чем больше, тем лучше. При этом желательно, чтобы его тексты по стилю не очень отличались от тех текстов, с которыми будет работать система. А если система должна работать с самыми разными текстами, то и корпус должен состоять из текстов самых разных.

Множество всех словоформ корпуса обозначим греческой буквой ν. Это множество может быть очень большим, но оно конечное — число слов в нем ограничено. Поэтому множество ν можно задать обычным перечислением.

(4) ν = {этот, большой, дракон, мальчик, увидел, мальчика, …}

Предложения в языке — это цепочки слов из множества ν. В конце предложения для технических нужд необходим символ, обозначающий конец предложения. В письменных текстах это обычно точка. Но для машины лучше использовать какое-то другое обозначение. Например, написать слово STOP.

Итак, предложения как цепочки слов могут быть, например, такие:

(5)
этот большой дракон увидел мальчика STOP
этот мальчик большой STOP
дракон этот этот STOP
дракон дракон дракон STOP

Каждому предложению, составленному из слов множества ν, необходимо приписать некоторую вероятность. Число предложений, которые можно построить из слов ν, не ограничено, поскольку длина предложения теоретически может быть какой угодно.

Как посчитать эту вероятность? Есть очень простой и очевидный способ. Считаем общее число предложений в корпусе. Допустим, их 10000. Затем для каждого предложения подсчитываем, сколько раз оно встретилось в нашем корпусе. Например, предложение *Мальчик увидел дракона* встретилось 5 раз. Делим 5 на 10000, получаем вероятность. Для предложения *Мальчик увидел дракона* вероятность равна 0,0005. Для остальных предложений из корпуса вероятность высчитывается точно так же. А для отсутствующих в корпусе предложений устанавливается вероятность, равная нулю.

У этого способа есть один важный недостаток. Оно приписывает нулевую вероятность всем предложениям, которые в нем не встретились.

Необходимо научиться оценивать вероятность для всех предложений, даже для тех, которые раньше нам никогда не встречались. Это можно сделать, анализируя фрагменты предложений — сочетания слов. При этом важно помнить, что вероятность появления слова зависит от того, какие слова появились перед ним. После слова *манная* скорее всего будет идти либо *каша,* либо *крупа,* либо *запеканка.* А если нам встретилось сочетание *съешь манную,* то за ним уже почти наверняка пойдет слово *кашу,* и ни на что другое надеяться не следует. *Съешь манную запеканку* в Интернете не встретилось.

Иными словами, нас интересует вероятность слова x_n, при условии, что перед ним идут слова $x_1, x_2 \ldots x_{n-1}$. Это можно записать с помощью формулы условной вероятности:

(6) $P(x_n \mid x_1, x_2 \ldots x_{n-1})$

Зная условную вероятность для каждого слова, мы можем посчитать вероятность любого предложения, состоящего из слов $x_1, x_2, x_3 \ldots x_n$. Для этого можно просто перемножить условные вероятности каждого отдельного слова:

(7)
$$P(x_1, x_2, x_3 \ldots x_n) = P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_1, x_2) \ldots \cdot P(x_n \mid x_1, x_2 \ldots x_{n-1})$$

Эта несколько упрощенная запись означает, что вероятность цепочки слов $x_1, x_2, x_3 \ldots x_n$ равна вероятности появления первого слова x_1, помноженной на условную вероятность появления второго слова x_2 (при условии, что ему предшествует слово x_1), помноженной на условную вероятность появления третьего слова x_3 (при условии, что предыдущие два слова — это x_1 и x_2), и так далее, до условной вероятности последнего слова x_n.

Однако вычислять такую длинную цепочку очень сложно, особенно учитывая, что для этого нам потребуется слишком большой корпус исходных данных. Но даже и в нем могут не встретиться какие-то необходимые нам длинные цепочки.

Чтобы избежать этого, надо либо сильно увеличить корпус, стараясь включить в него все возможные сочетания, либо изобрести какой-то другой способ вычисления вероятности.

Наиболее естественный альтернативный подход — опираться не на всю цепочку слов в предложении, а только на два-три предшествующих слова.

Этот метод был разработан в конце XX века, задолго до появления компьютеров, русским математиком А. А. Марковым. Он предположил, что не так важно, учитываем ли

мы всю длинную предшествующую цепочку или берем только один, два или три ближайших элемента — условная вероятность во всех этих случаях будет отличаться не очень сильно. Конечно, из этого обобщения бывают исключения. Но их мало, и для статистики они не очень важны.

Так появились **цепи Маркова** — статистические модели, позволяющие сильно упростить вычисление условной вероятности.

Цепь Маркова первого порядка самая грубая и самая простая. Она учитывает только одно предыдущее событие. Вероятность предложения на основе Марковской цепи первого порядка можно посчитать как вероятность первого слова, умноженную на вероятность второго слова при условии наличия первого слова, помноженную на вероятность третьего слова при условии наличия второго слова, и так далее, до вероятности последнего слова (которое у нас, кстати, всегда STOP) при наличии предпоследнего слова. Это можно записать с помощью формулы, которую мы, опять же, представим в несколько упрощенном виде, чтобы не утомлять читателя дополнительными объяснениями:

(8)
$$P(x_1, x_2, x_3 \ldots x_n) \approx P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_2) \ldots \cdot P(x_n \mid x_{n-1})$$

Цепь Маркова второго порядка чуть сложнее. При вычислении условной вероятности она учитывает не одно, а два предыдущих события. Впрочем, у первого события (слова) предыдущих событий нет, а у второго есть только одно, поэтому нормальный «второй порядок» начинается только с третьего события. Формально это выглядит вот так:

(9)
$$P(x_1, x_2, x_3 \ldots x_n) \approx P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_1, x_2) \ldots \cdot P(x_n \mid x_{n-2}, x_{n-1})$$

Возможны также цепи Маркова более высоких порядков, а также цепи для изменяющихся событий, но в основе моделей компьютерной обработки языка лежат обычно цепи первого и второго порядка.

3.3. Оценка максимального правдоподобия

Многие алгоритмы компьютерной обработки языка основаны на цепях Маркова второго порядка. Каждый множитель этой цепи состоит из трех идущих подряд элементов. В нашем случае это три идущих подряд слова. Сочетание из трех идущих подряд слов называют **триграммой** (trigram), а основанные на них модели — **триграммными языковыми моделями** (trigram language models).

Триграммная языковая модель включает в себя набор слов ν и набор параметров q (c |a, b), определенных для каждой возможной триграммы из ν, а также для сочетаний слов в начале и в конце предложения.

Параметр q (c |a, b) означает вероятность появления слова *c* при условии появления предшествующих слов *a* и *b*. При этом надо не забывать, что сочетание в конце предложения заканчивается на слово STOP, и в этом случае переменная *c* соответствует слову STOP. Например, параметр для самого последнего элемента в предложении *Вот дом, который построил Джек* будет выглядеть так: q (STOP |построил, Джек). Он соответствует вероятности того, что сочетание *построил Джек* окажется в предложении конечным.

В начале предложения первому слову ничего не предшествует — у него нет предыдущих слов. Можно обозначить это, введя специальный символ, например, звездочку, и заменить им переменные *a* и *b*. Тогда параметр для слова *Доколе* в начале

предложения, будет выглядеть вот так: q (Доколе |*, *). Он обозначает вероятность того, что слово *Доколе* является в предложении самым первым.

У второго слова есть только один предшественник — первое слово в предложении. Чтобы обозначить второго предшественника, опять воспользуемся звездочкой: q (ты |*, Доколе). Так записывается вероятность того, что если в начале предложения появилось слово *Доколе*, то сразу за ним последует слово *ты.*

Итак, триграммная модель языка включает в себя конечный набор слов v и набор параметров q (c |a, b), определенных для каждой возможной триграммы из этого набора, а также для сочетаний в конце и в начале предложения. Каждый параметр q (c |a, b) соответствует вероятности появления слова *c* после биграммы *ab*. И тогда вероятность появления предложения x_1, x_2, …. x_n вычисляется как произведение условных вероятностей для каждого слова этого предложения, включая конечный символ STOP. Это можно записать с помощью формулы:

(10)

$$P(x_1, x_2, x_3 \ldots x_n) = q(x_1 | *, *) \cdot q(x_2 | *, x_1) \cdot q(x_3 | x_1, x_2) \ldots \cdot q(x_n | x_{n-2}, x_{n-1})$$

Существует много разных путей для оценки параметров q (c |a, b). Наиболее естественный способ называется **оценкой максимального правдоподобия** (maximum-likelihood estimates). В корпусе можно подсчитать число биграмм *ab* и триграмм *abc*. Обозначим их как c(a, b) и c(a, b, c). И тогда вероятность появления слова *c* после биграммы *ab* равна числу триграмм *abc*, разделенному на число биграмм *ab*:

(11)

$$q_{ML}(c | a, b) = \frac{c(a, b, c)}{c(a, b)}$$

Допустим, сочетание *Доколе ты будешь* встречается в корпусе 10 раз. А сочетание *Доколе ты* — 15 раз. Тогда q (будешь | Доколе, ты) вычисляется так:

(12)

$$q(\text{будешь} | \text{Доколе, ты}) = \frac{c(\text{Доколе,ты,будешь})}{c(\text{Доколе,ты,})} = \frac{10}{15} = \frac{2}{3}$$

Если в корпусе после биграммы *кандидат филологических* всегда идет слово *наук*, то число биграмм *кандидат филологических* должно совпадать с числом триграмм *кандидат филологических наук,* что сделает результат деления равным единицы. Но это правильно: если после слова *кандидат филологических* всегда идет слово *наук*, то вероятность появления слова *наук* должна равняться единице.

У этого подхода, несмотря на его естественность, есть два серьезных недостатка. Во-первых, большинство потенциально возможных триграмм в корпусе не встретится, поэтому для подавляющего большинства теоретически возможных сочетаний условная вероятность окажется равной нулю. Во-вторых, если в корпусе не встретится какая-либо биграмма, значение c(a, b) для нее окажется нулевым, что даст ноль в знаменателе при оценке максимального правдоподобия.

Чтобы этих недостатков не было, необходимо найти такой способ оценки условной вероятности, который давал бы ненулевые значения даже в тех случаях, когда необходимые нам биграммы и триграммы в корпусе не встречаются. Это делается с помощью различных **методов сглаживания** (smoothed estimation methods), которые позволяют присваивать крошечные ненулевые вероятности для сочетаний, которые не встретились в корпусе.

3.4. Методы сглаживания

3.4.1. Линейная интерполяция

Линейная интерполяция (linear interpolation) — это простой метод сглаживания, использующий помимо триграмм также **биграммы** (bigram) и **униграммы** (unigram), то есть, цепочки из двух слов и отдельные слова.

Если триграмма в тренировочном корпусе не встретилась, то оценка максимального правдоподобия на основе этой триграммы будет равна нулю. Но, можно надеяться, что в корпусе встречались части этой триграммы — биграммы и униграммы. Поэтому оценку максимального правдоподобия можно проводить на основе сразу трех этих элементов: триграмм, биграмм и униграмм, заимствуя от каждой из них часть вероятности.

В предыдущем разделе мы определили оценку максимального правдоподобия на основе триграмм с помощью формулы (11). Оценка максимального правдоподобия на основе биграмм и униграмм выглядит следующим образом:

(13)

$$q_{ML}(c \mid b) = \frac{c(b, c)}{c(b)}$$

$$q_{ML}(c) = \frac{c(c)}{c()}$$

Здесь $q_{ML}(c \mid b)$ — это вероятность появления слова c после слова b, а $q_{ML}(c)$ — это вероятность появления слова *c* без учета контекста. В последней второй формуле c(c) обозначает число проявлений в корпусе униграммы (то есть, слова) *c*, а c() обозначает общее число слов в корпусе.

Оценка максимального правдоподобия на основе униграмм хороша тем, что она будет больше нуля для всех слов, которые хотя бы раз встретились в корпусе. Но она не учитывает контекст — она просто показывает вероятность появления того или другого слова самого по себе, в отрыве от других слов. Контекст учитывается при использовании триграмм.

Чтобы использовать достоинства одновременно триграмм и униграмм, а также находящихся между ними биграмм, можно взять понемногу от каждой из этих моделей. Это можно сделать, задав три переменных λ_1, λ_2 и λ_3, таких, что $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Тогда оценка максимального правдоподобия с учетом линейной интерполяции будет выглядеть так:

(14)

$$q(c \mid a, b) = \lambda_1 * q_{ML}(c \mid a, b) + \lambda_2 * q_{ML}(c \mid b) + \lambda_3 * q_{ML}(c)$$

Остается только придумать, как правильно подобрать значения λ_1, λ_2 и λ_3. Можно просто задать их как 1/3, тогда все три модели будут одинаковую роль в итоговом значении. Можно решить, что триграммы нам важнее, и сделать λ_1 большим. Есть также способы динамического вычисления трех параметров отдельно для каждого сочетания — в зависимости от числа биграмм, например.

3.4.2. Методы дисконтирования

Но как быть, если в тексте появляется сочетание, не встречавшееся в обучающем корпусе? Здесь поможет метод дисконтирования. Он предполагает, что мы заранее резервируем некоторую вероятность для не встретившихся в корпусе сочетаний. Для

этого необходимо отнять такую же точно вероятность от тех сочетаний, которые наблюдаются в корпусе. Как это делать?

Рассмотрим это на примере биграмм, а с триграммами дела обстоят точно так же. От числа появлений каждой биграммы надо отнять небольшое число β, так и накопится «запасная» вероятность, которую можно будет поделить между не встретившимися биграммами.

Пусть β=0,5, а часто именно так и бывает. Допустим, в нашем экспериментальном корпусе сочетание *купи еду* встретилось 15 раз, тогда *c*(*купи еду*)=15. Но мы посчитаем это как 15-β, то есть, c*(*купи еду*)=14,5. Точно так же, на 0,5, уменьшим показатели для всех других биграмм, начинающихся со слова *купи*. Результат можно записать в виде таблицы:

Таблица 1. Сглаживание вероятности с помощью метода дисконтирования

x	c(x)	c*(x)	$\frac{c^*(x)}{c(\text{купи})}$
купи еду	15	14,5	14,5/35
купи хлеба	10	9,5	9,5/35
купи поесть	5	4,5	4,5/35
купи соли	3	2,5	2,5/35
купи картошку	1	0,5	0,5/35
купи STOP	1	0,5	0,5/35
[ИТОГО] купи	35		

В последнем столбце Табл. 1 указана вероятность появления различных слов после *купи* с учетом дисконтирования. Какой объем вероятности у нас будет сэкономлен в результате этой операции? Нетрудно догадаться, что если каждый раз мы «откладывали» вероятность β, то сэкономить смоги β, умноженное на число различных видов биграмм. В нашем случае это 6 (шесть основных строк в таблице), так что мы сэкономили вероятность $6\times\beta=6\times 0{,}5 = 3$.

Останется только поделить эту сэкономленную вероятность между всеми теоретически возможными биграммами, начинающимся со слова *купи,* которые не встретились в обучающем корпусе.

3.5. Модель перевода

Предложение можно перевести с одного языка на другой разными способами. В некоторых случаях перевод может быть очень странным. Так, английское предложение *Masha ate sushi* большинство переведет на русский как *Маша ела суши*. Но кто-то, возможно, переведет его как *Маша ела кашу.* Перевел ведь Набоков «Алису в стране чудес» как «Аня в стране чудес». Теоретически нельзя отрицать, что кто-то переведет это предложение как *Маша ел каша* или же как *Суши каша Маша*. У каждого из этих вариантов перевода есть своя вероятность, хотя у всех вариантов в этом примере кроме первого она близка к нулю.

Модель перевода (translation model) позволяет оценить вероятность того, что одно предложение является переводом другого. Или, если быть точнее, вероятность того, что предложение *x* — это исходное предложение для переводного предложения *y*. Вспомним формулу (3) в разделе 3.1 — нас интересует *p(x|y),* где *x* — исходное предложение (вероятность которого оценивается), а *y* — это переведенное предложение (которое рассматривается как имеющееся условие). Именно эти вероятности и оценивает модель перевода.

3.5.1. Выравнивание

Для создания модели перевода нужны параллельные корпусы. Если у нас есть большие тексты и их переводы на другой язык, то это еще не параллельный корпус. В корпусе должны быть установлены связи между элементами исходных текстов и из переводов. Эти связи называются **выравниванием** (alignment), и они могут быть разного уровня. Бывает выравнивание по документам, по абзацам, по предложениям и по словам. Иногда встречаются и более экзотические виды выравниваний, но для создания автоматического переводчика будет достаточно параллельного корпуса, в котором есть **выравнивание по предложениям** (sentence alignment), то есть, где каждое предложение связано со своим переводом. Именно эти корпусы наиболее популярны.

На первый взгляд может показаться, что каждое отдельное предложение в исходном тексте переводится на другой язык одним предложением, что можно обозначить как 1:1. Но такие случаи составляют в среднем около 90% [Manning and Schütze 1999:468]. Остальные 10% вызывают проблему соответствия. При переводе два коротких предложения могут объединиться в одно, что можно обозначить как 2:1. Возможны и другие варианты, например, 1:2, 1:3, 3:1 и даже 2:2, если два предложения при переводе были перемешаны и разбиты на два других предложения. Иногда фрагменты текста в переводе могут переставляться, так что порядок следования предложений меняется, тогда возможны более сложные сочетания.

Выравнивание можно выполнять вручную, но это очень медленный процесс, особенно для больших корпусов. Поэтому обычно выравнивание текстов производится автоматически или полуавтоматически.

Для автоматического выравнивания существуют разные алгоритмы. В основе одного из них лежит простое предположении о том, что длинные фрагменты текста переводятся длинными фрагментами, а короткие — короткими. Длина фрагментов измеряется в словах или в символах. Этот способ часто применяют для выравнивания абзацев, после которого производится выравнивания предложений внутри абзацев. Хотя при этом игнорируется вся лексическая информация, этот подход дает довольно хорошие результаты. Другие методы выравнивания предполагают использование небольших двуязычных словарей. Если слово встречается и в исходном предложении, и в переводе, это становится дополнительным доводом в пользу установления связи между этими предложениями. Еще одним доводом может стать совпадение чисел и дат.

Большинство моделей статистического перевода используют выравнивание по словам. Рассмотрим несколько подробнее этот вид выравнивания.

Допустим, у нас есть следующее предложение на русском языке:

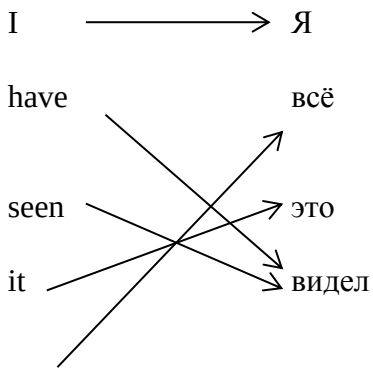
y = *Я всё это видел*

Предположительно, оно является переводом с английского предложения:

x = *I have seen it all*

Выравнивание этих предложений по словам можно изобразить следующим образом:

(15)



all

Эта схема записывается с помощью последовательности a_1, a_2 … a_m, число элементов которой равно числу слов в исходном предложении *x*. В нашем случае это английское предложение, в нем пять слов. Значит, у нас получится массив выравнивания из пяти элементов, по одному для каждого английского слова. Значение соответствующего элемента равно номеру русского слова, с которым связано соответствующее английское слово. В нашем случае массив получается такой:

(16)
$a_1, a_2, a_3, a_4, a_5 = <1, 4, 4, 3, 2>$

В этом массиве может встретиться ноль, который возникает в тех случаях, когда какое-нибудь слово в исходном предложении не имеет аналога в переводе. Например, это относится к английским вспомогательным глаголам.

Большой набор бесплатных параллельных корпусов доступен в проекте OPUS [Tiedemann and Nygaard 2004].

Самый известный алгоритм для выравнивания текстов по предложениям называется Hunalign. Он объединяет 2 подхода: использование длины предложений и выравнивание со словарём [Varga et al., 2005]. На его основе действует удобная утилита LF Aligner.

Для пословного выравнивания параллельного корпуса чаще всего используется программа GIZA++. Она основана на моделях IBM, о которых мы поговорим в следующем разделе.

3.5.2. Модели перевода, основанные на словах

Первые попытки построить модель перевода были связаны с анализом отдельных слов, поэтому их общепринятое название — **модели перевода, основанные на словах** (word-based translation models). В этой области классическими стали подходы, разработанные в компании IBM, которые так и называются: модель IBM-1, модель IBM-2, и так далее до модели IBM-5. О них подробно рассказывается, например в курсе Михаэля Коллинза на Coursera и в учебнике [Koehn 2010].

Рассмотрим, как устроены эти модели. Напомню, что мы ищем значение p(x|y). Это «обратная вероятность», то есть вероятность того, что предложение *x* является исходным по отношению к конечному предложению *y*.

В предыдущем разделе мы анализировали английское предложение *I have seen it all* и его возможный перевод на русский язык *Я всё это видел*. Значит, мы пытаемся оценить условную вероятность p (I have seen it all | Я всё это видел).

Попробуем записать формулу вероятности несколько подробнее. Допустим, конечное предложение *y* состоит из *k* слов: $y_1, y_2 \ldots y_k$, а исходное предложение *x*, вероятность которого мы ищем, состоит из *m* слов: $x_1, x_2 \ldots x_m$. Тогда нас интересует значение вероятности $p(x_1, x_2 \ldots x_m | y_1, y_2 \ldots y_k)$.

Но посчитать такую вероятность напрямую очень сложно. Многие модели упрощают эту формулу необычным способом — в нее добавляют дополнительные параметры. Это длина исходного предложения *m* и выравнивание по словам $a_1, a_2 \ldots a_m$. Длина исходного предложения *m* считается известной, потому что мы оцениваем вероятность для конкретного предложения $x_1, x_2 \ldots x_m$.

Тогда формула искомой вероятности будет выглядеть так:

(17) $p(x_1\, x_2 \ldots x_m, a_1\, a_2 \ldots a_m \mid y_1\, y_2 \ldots y_k, m)$

Опишем словами, что обозначает формула (17). Это условная вероятность. У нас есть предложение-перевод, которое состоит из последовательности слов $<y_1, y_2 \ldots y_k>$, известна также длина исходного предложения — *m* слов. Это условия для вычисления условной вероятности того, что исходное предложение состоит из последовательности *m* слов: $<x_1, x_2 \ldots x_m>$, и что его выравнивание по словам с предложением x описывается последовательностью $<a_1, a_2 \ldots a_m>$.

Если исходное английское предложение *I have seen it all* состоит из пяти слов, то m=5. Его возможный перевод на русский язык — *Я всё это видел*. Выравнивание этих предложений задается массивом $a_1, a_2, a_3, a_4, a_5 = <1, 4, 4, 3, 2>$.

Подставим эти значения в нашу формулу. Нас интересует следующая условная вероятность:

(18) p(I have seen it all, 1 4 4 3 2 | Я всё это видел, 5)

Но как научиться считать эту вероятность? Один из способов предлагает модель IBM-2. Она упрощает формулу условной вероятности следующим образом:

(19)

$$p(x_1\, x_2 \ldots x_m, a_1\, a_2 \ldots a_m \mid y_1\, y_2 \ldots y_k, m) = \prod_{i=1}^{m} q(a_i \mid i, k, m) \times t(x_i \mid y_{a_i})$$

Рассмотрим подробнее последовательность упрощений в (19), поскольку подобные приемы используются во многих статистических моделях языка.

Первое упрощение связано со свойствами условной и совместной вероятности. Это свойство можно записать следующим образом:

(20)
p(A B| условия) = p(B| условия) × p(A| B, условия)

Перепишем по этой схеме формулу (17):

(21)

$$p(x_1\, x_2 \ldots x_m, a_1\, a_2 \ldots a_m \mid y_1\, y_2 \ldots y_k, m) =$$

$$= p(a_1 \ldots a_m \mid y_1 \ldots y_k, m) \times p(x_1 \ldots x_m \mid a_1 \ldots a_m, y_1 \ldots y_k, m)$$

Теперь рассмотрим каждую из двух составляющих в этой формуле отдельно. Начнем с первого множителя:

(22)

$$p(a_1 \ldots a_m \mid y_1 \ldots y_k, m) = p(a_1 \mid y_1 \ldots y_k, m) \times p(a_2 \mid a_1, y_1 \ldots y_k, m) \times$$
$$\times p(a_3 \mid a_1\, a_2, y_1 \ldots y_k, m) \times \ldots \times$$
$$\times p(a_m \mid a_1 \ldots a_{m-1}, y_1 \ldots y_k, m)$$

$$= \prod_{i=1}^{m} p(a_i \mid a_1 \dots a_{i-1}, y_1 \ \dots \ y_k, m)$$

Так раскладывается на множители первый элемент в формуле (21). Осталось только придумать, как вычислять для всех *i* элементы $p(a_i \mid a \dots a_{i-1}, y_1 \dots y_k, m)$. Модель IBM-2 и здесь сильно упрощает ситуацию. Она предлагает рассматривать их как параметры системы и считать по формуле $q(a_i \mid i, k, m)$:

(23)

$$\prod_{i=1}^{m} p(a_i \mid a_1 \dots a_{i-1}, y_1 \ \dots \ y_k, m) \approx \prod_{i=1}^{m} q(a_i \mid i, k, m)$$

Вспомним, что означают элементы параметра $q(a_i \mid i, k, m)$. В правой его части записаны условия: длина исходного предложения *m*, длина конечного предложения *k*, а также номер слова *i* в исходном предложении. При этих условиях мы вычисляем значение переменной a_i, которая соответствует номеру того слова в переводе, которое связано со словом номер *i* в исходном предложении.

Другими словами, модель IBM-2 учитывает вероятность того, что слово номер *i* в исходном предложении связано со словом номер a_i в переводе, а длины предложений при этом равны, соответственно, *m* и *k*. Состав предложений при этом значение не имеет.

Теперь рассмотрим вторую составляющую в формуле (21): $p(x_1 \dots x_m \mid a_1 \dots a_m, y_1 \dots y_k, m)$.

Ее тоже можно переписать:

(24):

$$\begin{aligned} p(x_1 \dots x_m \mid a_1 \dots a_m, y_1 \dots y_k, m) &= p(x_1 \mid a_1 \dots a_m, y_1 \dots y_k, m) \times \\ &\times p(x_2 \mid x_1, a_1 \dots a_m, y_1 \dots y_k, m) \times \dots \times \\ &\times p(x_m \mid x_1 \dots x_{m-1}, a_1 \dots a_m, y_1 \dots y_k, m) \\ &= \prod_{i=1}^{m} p(x_i \mid x_1 \dots x_{i-1}, a_1 \dots a_m, y_1 \dots y_k, m) \end{aligned}$$

Получившееся произведение довольно сложное, и модель IBM-2 его тоже очень сильно упрощает. Она убирает из условий все значения *a*, все значения предыдущих *x* и все значения *y* кроме того, которое по выравниванию связано с x_i. Остается такой параметр: $t(x_i \mid y_{ai})$

Таким образом, подставив оба параметра в систему, мы получаем следующую формулу:

(25):

$$p(x_1 x_2 \dots x_m, a_1 a_2 \dots a_m \mid y_1 y_2 \dots y_k, m) = \prod_{i=1}^{m} q(a_i \mid i, k, m) \times t(x_i \mid y_{a_i})$$

Если мы хотим оценить вероятность того, что английское предложение *I have seen it all* — это перевод с русского оригинала *Я всё это видел* с упомянутым выше выравниванием <1 4 4 3 2>, то по формуле (25) нам надо посчитать следующее выражение:

(26)

$$p(\text{I have seen it all, 1 4 4 3 2} \mid \text{Я всё это видел, 5}) =$$
$$q(1 \mid 1, 4, 5) \times t(\text{I} \mid \text{Я})$$
$$\times q(4 \mid 2, 4, 5) \times t(\text{have} \mid \text{видел})$$
$$\times q(4 \mid 3, 4, 5) \times t(\text{seen} \mid \text{видел})$$
$$\times q(3 \mid 4, 4, 5) \times t(\text{it} \mid \text{это})$$
$$\times q(2 \mid 5, 4, 5) \times t(\text{all} \mid \text{всё})$$

Осталось только узнать, чему эти параметры равны.

3.5.3. Как считать параметры

Чтобы модель IBM-2 работала, необходимо научиться вычислять два основных параметра этой модели: q(j | i, k, m) и t(x | y). Точнее, приходится говорить не о вычислении этих параметров, а об их примерной оценке на основе имеющегося у нас параллельного корпуса.

Представим вначале идеальную ситуацию. У нас есть большой размеченный параллельный корпус. Выравнивание в нем проведено не только по предложениям, но и по словам. С помощью такого идеального корпуса параметры оцениваются легко и естественно. Рассмотрим, как это происходит.

Если у нас есть параллельный размеченный корпус, то можно его представить в виде тройки значений $(x^{(l)}, y^{(l)}, a^{(l)})$. Количество этих троек равно числу переведенных фрагментов, которое обозначим как n. Мы будем называть их предложениями, но они могут состоять и из других фрагментов текста. Рассмотрим каждый элемент этой тройки отдельно.

$x^{(p)}$ — это набор исходных предложений $x^{(1)}$, $x^{(2)}$… $x^{(n)}$, каждое их них состоит из m_p слов: $x^{(p)}_1\ x_2^{(p)}\ \ldots\ x_{m_p}^{(p)}$, при этом p здесь обозначает порядковый номер предложения в корпусе.

$y^{(p)}$ — это набор конечных предложений, переводов. Их тоже n. Каждое переведенное предложение состоит из k_p слов: $y^{(p)}_1\ y^{(p)}_2\ \ldots\ y^{(p)}_{k_p}$.

$a^{(p)}$— это массив выравнивания, в нем столько же элементов, сколько в исходном предложении: $a^{(p)}_1\ a^{(p)}_2\ \ldots\ a^{(p)}_{m_p}$. Напомним, что в массиве выравнивания $a^{(p)}$ хранятся указания на то, с каким словом в конечном предложении связано определенное слово в исходном предложении. Например, если $a^{(239)}_2=3$, это означает, что в исходном предложении номер 239 второе слово переведено с помощью третьего слова в конечном предложении.

Если на материале корпуса определены такие тройки, то параметры считаются очень просто. Для этого компьютер должен просмотреть каждую тройку и посчитать следующие значения:

- сколько раз в исходном корпусе встретилось каждое слово — значение $c(x)$;
- сколько раз это слово было переведено с помощью слова y — обозначим его как $c(x \blacktriangleright y)$;
- сколько раз слово за номером i в исходном предложении было связано со словом номер j в конечном предложении, если длины предложений, соответственно, m и k — значение $c(j \mid i, m, k)$;
- сколько раз вообще позиция i встречалась в исходном предложении, если длины исходного и конечного предложений были равны, соответственно, m и k — значение $c(i, m, k)$.

После того, как на материале корпуса будут сосчитаны все эти значения, параметры модели вычисляются по принципу оценки максимального правдоподобия:

(27)

$$q(j \mid i,k,m) = \frac{c(j \mid i,m,k)}{c(i,m,k)} \qquad t(x \mid y) = \frac{c(x \blacktriangleright y)}{c(x)}$$

При обработке корпуса просматривается каждая тройка, для которой выполняются однотипные операции:

1) для каждой тройки просматриваются все слова в переведенном предложении (для всех i от 1 до k_p);
2) для каждого из них просматривается каждое слово в исходном предложении — начиная с нулевого слова (для всех j = 0 до m_p; напомним, ноль в массиве выравнивания возникает в тех случаях, когда какое-нибудь слово в исходном предложении не имеет аналога в переводе);
3) если $a^{(p)}_i$ = j, то на единицу увеличиваются следующие значения:
   $c(x^{(p)}_j \blacktriangleright y^{(p)}_i)$
   $c(x^{(p)}_j)$
   $c(j \mid i, m_p, k_p)$
   $c(i, m_p, k_p)$
4) Когда весь корпус обработан, значения параметров вычисляются по формулам (27)

Рассмотрим это на примере. Допустим, *No money* — это предложение номер 239 в исходном корпусе. Его перевод — русское предложение Д*енег нет.*

$x^{(239)}$ = *No money*, m_{239}=2

$y^{(239)}$ = Д*енег нет,* k_{239}=2

$a^{(239)}$ = <2, 1>, значит, $a^{(239)}_1$ = 2, $a^{(239)}_2$ = 1 (первое слово в исходном предложении соответствует второму в переводе, а второе — первому).

Всего получается 6 вариантов, так мы рассматриваем *i* от 1 до 2 и *j* от 0 до 2. Для каждого сочетания i и j нас интересует, выполняется ли равенство $a^{(239)}_i$ = j

Запишем результаты в форме таблицы.

Таблица 2. Пример анализа параметров на размеченном корпусе

i	j	$a^{(239)}_i$	$a^{(239)}_i$ = j
1	0	2	нет
1	1	2	нет
1	2	2	да
2	0	1	нет
2	1	1	да
2	2	1	нет

Последняя колонка в таблице 2 показывает, в каких случаях, $a^{(239)}_i$ = j. Это две пары: (i=1, j=2) и (i=2, j=1). Соответственно, на единицу увеличатся следующие счетчики:

Таблица 3. Результата анализа параметров таблицы 2

Для (i=1, j=2)	Для (i=2, j=1)

c(нет, no)	c(денег, money)
c(нет)	c(денег)
c(2 | 1, 2, 2)	c(1 | 2, 2, 2)
c(1, 2, 2)	c(2, 2, 2)

Описанный подход годится только для идеального случая, когда у нас есть большой параллельный корпус, выравненный по словам. К сожалению, в жизни обычно не попадаются полностью размеченные данные. Как правило, в параллельном корпусе существует выравнивание только по предложениям, без выравнивания по словам, поэтому массива $a^{(n)}$ у нас нет. Как быть в этой ситуации?

Здесь на помощь приходит **EM-алгоритм**. По-английски он называется Expectation-maximization (EM) algorithm, что можно перевести как **алгоритм увеличения ожидания**, но более точно его можно назвать **алгоритмом увеличения ожидаемого правдоподобия**. Он позволяет вычислять оценку максимально правдоподобия в тех случаях, когда часть данных неизвестна.

EM-алгоритм действует следующим образом. Вначале всем параметрам присваиваются произвольные значения. Затем на основе этих параметров несколько раз подряд обрабатывает имеющийся корпус, после каждой обработки параметры корректируются. Обычно корпус нужно обработать 10-20 раз, чтобы значения искомых параметров почти перестали изменяться — результат получается достаточно приемлемым, хотя корпус не был размечен.

Получается, что с помощью EM-алгоритма можно вычислять параметры модели, не имея массива выравнивания. Но в таком случае, получив необходимые параметры модели, можно вычислить для каждой пары предложений ее массив выравниваний. Иными словами, с помощью моделей пословного перевода и EM-алгоритма можно проводить выравнивание по словам — на этом основана работа популярной программы GIZA++, которая выравнивает предложения по словам. Но для перевода пословные модели оказались неэффективными. Им на смену пришли модели переводов на основе сочетаний (фразовых таблиц).

3.5.4. Перевод на основе сочетаний

В 1990-е годы начал развиваться новый подход к машинному переводу, использующий цепочки слов — фрагменты предложений. Так появились **модели перевода, основанные на сочетаниях** (phrase-based translation models).

Важно сказать, что сочетания в этих моделях не соответствуют словосочетаниям в лингвистическом понимании этого слова. Сочетания в машинном переводе — это фрагменты предложений, не имеющие единой структуры, например, фрагмент *пришел в этот*.

В основе этого подхода лежат тройки (x, y, g), где x — фрагмент предложения на исходном языке, y — фрагмент на языке перевода, а g — вес пары (x, y), который тем выше, чем чаще встречается перевод одного фрагмента другим. Приведем примеры таких троек:

(in the, в, 0,6)
(in the town, в городе, 0,007)
(work in the town, работают в городе, 0,0006)

Обратите внимание, что количество слов в первом и во втором фрагменте может отличаться.

Для автоматического создания таких троек нам требуется параллельный корпус из предложений на двух языках $x^{(n)}$ и $y^{(n)}$, а также **матрица выравнивания** (alignment matrix) по словам $A^{(n)}$. Матрица — математический объект, похожий на прямоугольную таблицу, в клетки которой вписаны различные значения. В нашем случае в эти клетки будут вписаны единицы или нули, обозначающие выравнивание слов в предложениях, записанных по вертикальной и горизонтальной стороне прямоугольника.

Матрица выравниваний строится на основе двух одномерных выравниваний, выполненных с помощью EM-алгоритма. Вначале мы создаем массив выравнивания, рассматривая первый язык как оригинал, а второй — как перевод. Затем мы запускаем тот же алгоритм создания массива выравнивания, поменяв местами перевод и оригинал. Получается другая одномерная версия возможного выравнивания. Далее, мы заносим полученные результаты в массив. В каких-то клетках они совпадут — и это значит, что связь или отсутствие связи между этими словами почти не вызывает сомнений. Но где-то результаты будут отличаться, тогда для какой-то клетки один массив выравниваний будет показывать единицу, а второй — ноль. Для этих случаев есть свои правила, и после их применения получается достаточно правдоподобная матрица для каждой пары предложений.

Вот так будет выглядеть, например, матрица выравнивания для английского предложения *She will go shopping* и для его русского перевода *Она пойдет в магазин*. Мы запишем здесь только единицы, а нули опустим.

	She	will	go	shopping
Она	1			
пойдет		1	1	
в				1
магазин				1

Обратите внимание, что матрица позволяет передавать те связи, которые невозможно изобразить с помощью одномерного массива из моделей IBM. В нашем примере два английских слова *will go* передаются одним русским словом *пойдет*, но при этом слово *shopping* передается двумя русским словами *в магазин*. Это невозможно передать, если в массиве $a^{(n)}$ число элементов совпадает с числом слов в одном их предложений. Какие-то из связей просто не поместятся в одномерный массив.

Теперь нам надо узнать, как на основе параллельного корпуса и матрицы выравниваний создать тройки типа *(in the town, в городе, 0,007)*, на основе которых будет работать машинный перевод. Для этого нужно выполнить следующие шаги:

1. выделить в исходном и в конечном предложении все возможные фрагменты;
2. проверить, какие пары фрагментов являются консистентными — это мы поясним чуть ниже;
3. для консистентных пар посчитать вес, тогда два фрагмента и вес образуют необходимую тройку.

Консистентность можно описать с помощью трех условий:

а) во фрагменте $[y_v, y_w]$ есть хотя бы одно слово, связанное со словом во фрагменте $[x_i, x_j]$;

б) ни одно слово во фрагменте $[x_i, x_j]$ не связано со словом за пределами $[y_v, y_w]$;

в) ни одно слово во фрагменте $[y_v, y_w]$ не связано со словом за пределами $[x_i, x_j]$.

Консистентность можно легко увидеть с помощью матрицы выравнивания. Если два слова связаны, то соответствующей клетке матрицы стоит единица. Все единицы двух консистентных фрагментов можно обвести прямоугольником, за пределами которого не останется единиц ни по горизонтали, ни по вертикали (это соответствует условиям *б* и *в*).

Вот примеры двух таких треугольников, они соответствуют двум парам отрезков: *(She will go, она пойдет)* и *(shopping, в магазин).*

	She	will	go	shopping
Она	1			
пойдет		1	1	
В				1
магазин				1

Для двух этих предложений можно найти и другие консистентные пары, например: *(will go shopping, пойдет в магазин).*

Получившиеся тройки образуют **словарь сочетаний** (phrase-based lexicon), на основе которого будет происходить перевод предложений.

После этого по определенному алгоритму необходимо вычислить вес этих пар. Далее с помощью некоторой громоздкой формулы можно будет вычислять модель перевода. Но мы не будем описывать здесь дальнейшие шаги по созданию модели перевода. Если предыдущая часть показалась вам интересной и полезной, то дальнейшие шаги вы можете узнать из классической книги по статистическому машинному переводу [Koehn 2010], из лекций Михаэля Коллинза на Coursera и во многих специализированных пособиях и учебниках.

Самая удобная система для реализации статистического машинного перевода называется Moses. Она объединяет все необходимые для этого программы, алгоритмы и утилиты, но требует больших ресурсов.

4. Гибридный перевод

Компьютерные лингвисты, разрабатывающие системы на основе правил, давно поняли, что статистика тоже может им пригодиться. Например, синтаксический анализ известного предложения *Time flies like an arrow* может дать два разных варианта: *Время летит как стрела* и *Мухи времени любят стрелу.* Более удачный перевод можно выбрать с помощью статистики — посмотреть, например, как часто встречаются сочетания *время летит* и *мухи времени*, а также *летит как стрела* и *любят стрелу*.

Что касается математиков, специалистов по статистике, то они часто считают ненужной любую лингвистическую информацию. Для достижения наибольшей эффективности (а может быть, для примирения математиков и лингвистов), большинство систем машинного перевода сегодня используют и правила, и статистику. Такой подход называется гибридным.

Объединять правила и статистику можно разными способами. Можно интегрировать статистический модуль в перевод, основанный на правилах — именно это мы описали, выбирая более удачный перевод предложения *Time flies like an arrow*.

Другой подход связан с интеграцией правил в статистическую модель. Как пример тут можно привести перевод с японского языка на русский или на английский. В японском языке сказуемое обычно находится в конце предложения. Такой порядок слов сильно отличается от привычного нам порядка *Маша ела кашу* — подлежащее-сказуемое-дополнение. Это отличие может создать большие неудобства при формировании фразовых таблиц модели перевода. Ведь фразовая таблица состоит из параллельных фрагментов, а русское сочетание *Маша ела* может протянуться от начала до конца японского предложения: подлежащее в самом начале, а сказуемое в самом конце.

Чтобы сделать японские фразы более «похожими» на русские, можно перед применением статистики провести предобработку японского текста с помощью правил —

просто найти в каждом предложении сказуемое и переместить его поближе к подлежащему. Конечно, получившийся текст будет уже не японским, а псевдо-японским, но перевести его на русский будет гораздо легче.

Интересный вариант гибридной технологии используется в компании PROMT. Там создан большой обучающий параллельный корпус «исправленных ошибок». Он состоит из предложений, переведенных системой с помощью правил, в соответствие которым поставлены эти же предложения, исправленные носителями языка. На основе этого корпуса работает модуль синтаксического постредактирования — система обучена переводить «с русского на русский». Таким образом, перевод происходит в два этапа: (1) перевод по правилам; (2) статистическая доработка переведенного текста с помощью системы, обученной на параллельном корпусе исправлений [Молчанов 2013].

Практически все разработчики систем машинного перевода используют сейчас гибридные технологии, но исследования в этой области защищены коммерческой тайной. В мире пока нет ни одного учебника по гибридным технологиям, какие-то отдельные разработки иногда описываются в выступлениях на конференциях.

Но будущее прикладной лингвистики и, в частности, машинного перевода, связано именно с гибридными технологиями. Скорее всего, наиболее эффективные системы будут действовать на основе статистики, усиленной правилами.

5. Методы оценки качества перевода

Оценивать качество перевода можно с помощью экспертов или автоматически, без участия людей.

Для оценки с помощью экспертов нужны эксперты. В 1960-е года американская комиссия ALPAC пришла к выводу, что их должно быть не менее четырех человек.

Для экспертной оценки существует несколько методик. Например, перевод каждого предложения эксперты могут оценивать по двум параметрам: **полнота** (adequacy) и **гладкость** (fluency). Полнота отвечает за точность перевода, а гладкость — за правильность фразы с точки зрения носителя языка. По каждому из этих параметров каждый эксперт ставит оценки переводам по заранее заданной шкале, например, от 1 до 4.

Другой способ экспертной оценки — это выбор лучшего перевода из нескольких или ранжирование имеющихся переводов. Еще один подход связан с измерением времени и сил, затраченных человеком-редактором на исправление того или иного автоматического перевода.

В любом случае, экспертный подход требует больших трудозатрат, а оценить с его помощью можно лишь небольшое число предложений. Гораздо удобнее в этом отношении автоматическая оценка перевода.

Автоматическая оценка предполагает сравнение переведенного текста с эталонным переводом. В качестве эталона может использоваться перевод, сделанный людьми, или отредактированный текст машинного перевода. Чем ближе результат к эталону, тем качественнее машинный перевод. Это метод довольно грубый, но его удобно применять при оценке больших объемов текста.

В основе автоматической оценки перевода должна лежать определенная метрика, то есть способ числовой оценки различий между переводом и эталоном. Самая известная метрика называется BLEU (BiLingual Evaluation Understudy) [Papineni et al, 2002], предложенная компанией IBM в 2002 году. Она учитывает, сколько n-грамм совпадает в переводе и в эталоне, а затем по определенной формуле выводит оценку качества перевода по шкале от 0 до 100.

Существуют и другие метрики оценки качества перевода: NIST, MERT, METEOR, TER. Посмотреть их работу, а заодно и оценить качество перевода различных переводчиков можно помощью бесплатной онлайн системы Asiya.

При создании машинных переводчиков необходимо производить оценку качества после каждого серьезного изменения системы. Можно, например, автоматически отслеживать, какие предложения стали переводиться иначе, а затем в каждом отдельном примере оценивать изменение перевода: стала ли новая версия лучше или хуже. Если процент ухудшений достаточно велик, изменение системы можно признать неудовлетворительным.

6. Некоторые современные системы машинного перевода

В этом разделе мы рассмотрим основные системы машинного перевода. Почти все они предоставляют бесплатные онлайн-версии, с помощью которых читатель сможет сравнить их работу.

Systran (США, затем Франция,затем Корея)

Это самая старая из успешных и самая успешная из старых переводческих систем. Американская компания Systran начала разрабатывать ее в 1968 году для Военно-воздушных сил США. Первым делом был создан переводчик с русского на английский — шла холодная война. В 1973 году к нему добавился переводчик с английского на русский, его разрабатывали специально для совместного космического проекта СССР и США «Союз-Аполлон». В 1986 году компанию Systran купила французская семья Гашо (Gachot), с тех пор Systran считался французской системой. Но в 2014 году ее контрольный пакет акций приобрела корейская фирма CSLi, которая, в частности, создавала переводчик для последних моделей Samsung Galaxy. Компания CSLi изменила свое название на Systran International и планирует стать абсолютным лидером машинного перевода.

Переводчик Systran используется во многих других компаниях, например, на его основе действует перевод Babel Fish в системе Yahoo! До 2007 года на основе Systran работал переводчик Google. Клиентом Systran до сих пор является Министерство обороны США.

Systran предоставляет бесплатный онлайн-перевод на своем сайте http://www.systransoft.com/, там же продаются его системы различных конфигураций. Перевод разработан для 52 языков, бесплатная онлайн-версия доступна только для десяти из них.

Logos и OpenLogos (США и Германия)

Переводчик Logos начал разрабатываться компанией Logos Corporation в 1970 году специально для войны во Вьетнаме, поэтому первая пара языков Logos была англо-вьетнамской. Рабочая версия для этой пары была представлена в 1972 году, за три года до окончания вьетнамской войны, которая длилась более 19 лет. Коммерческая версия программы сейчас разрабатывается компанией Group Business Software AG для операционной системы Microsoft Windows.

С 2005 года Немецкий исследовательский центр по искусственному интеллекту (DFKI) создает бесплатную версию этого переводчика с названием OpenLogos. Эта система переводит с немецкого и английского языков на французский, итальянский, испанский и португальский. Она работает на основе правил и легко позволяет подключать новые словари.

PROMT (Россия)

Это лучший российский переводчик. Он разрабатывается компанией PROMT, главный офис которой находится в Санкт-Петербурге. Компании принадлежит первый онлайн-переводчик русскоязычного интернета.

PROMT был создан в 1991 году сотрудниками лаборатории инженерной лингвистики Ленинградского педагогического института им. А. И. Герцена. Уже в 1992 году компания

выиграла тендер NASA на поставку систем машинного перевода с английского языка на русский. С 1992 по 1998 год эта программа называлась STYLUS. Сейчас разработаны переводчики PROMT для 64 языковых пар, в том числе для 34 пар без использования русского языка.

Вначале PROMT действовал только на основе правил, но с 2010 года система стала использовать также статистические и гибридные технологии. Гибридные технологии PROMT используются в работе международного туристического онлайн-сервиса TripAdvisor.

В 2013 и 2014 годах компания становилась победителем в конкурсе разработчиков систем машинного перевода в рамках Семинара по статистическому машинному переводу, который проводится под эгидой Ассоциации компьютерной лингвистики (ACL).

Linguatec (Германия)

Эта система разрабатывается с 1996 года немецкой компанией Linguatec, которая специализируется также на синтезе речи и преобразовании устной речи в текст. Иными словами, Linguatec занимается не только переводом, но также учит машину читать вслух тексты и записывать то, что ей диктует человек. Переводчик Linguatec основан на правилах, в 2006 году компания запатентовала технологию нейронного трансфера, работающего по принципам нейронной сети, с 2010 года в систему стали интегрироваться гибридные технологии.

IdiomaX (Швейцария, но больше похоже на Италию)

Этот переводчик с 1996 года разрабатывается швейцарской компанией IdiomaX. С 1998 года в течение нескольких лет итальянская компания Garzanti распространяла этот переводчик в книжных магазинах Италии, поэтому особой популярностью он пользуется у итальянцев. Действует на основе правил.

Babylon (Израиль)

Разрабатывается израильской компанией Babylon с 1997 года. Первые версии были бесплатными, но позже эта система стала платной. В 2011 году программа стала лидером по числу бесплатных скачиваний и была занесена в Книгу рекордов Гиннеса. Число пользователей программы превысило 100 миллионов, сайт компании попал в список 100 наиболее посещаемых сайтов мира. Но в 2012 компоненты системы были признаны вредоносными, поскольку переводческая панель Babylon устанавливала в браузере свою собственную главную страницу и страницу поиска, не позволяя их переустанавливать.

Apertium (Испания)

Эту систему по заказу испанского правительства начали создавать четыре университета и три лингвистических компании. Первая версия Apertium была представлена миру в 2005 году. Это открытая система, основанная на правилах. На сайте SourceForge можно бесплатно скачать весь код Apertium и ее лингвистические данные. Система создавалась вначале для близкородственных языков, но сейчас она работает для самых разных пар. Чтобы подключить новую языковую пару, нужно добавить словарь и лингвистическую информацию о структуре необходимых языков.

Google Переводчик, Google Translate (США)

Бесплатный переводческий сервис Google. Предлагает перевод для любой языковой пары из 73 языков — например, перевод с телугу на хмонг.

Этот переводчик разработан американской компанией Google. Вначале она предлагала перевод на основе системы Systran. Но в 2004 году руководство компании решило отказаться от Systran и создать свой собственный переводчик на основе статистических

методов. В апреле 2006 фирменный переводчик Google был запущен. Он действовал для пар арабский-английский и английский-арабский, но скоро к нему прибавились пары с русским и китайским языком. В 2007 году Google перевел все имевшиеся у него языковые пары на статистические методы перевода, отказавшись от использования переводчика Systran. После этого в систему добавлялись новые языки и опции.

Переводчик Bing (США)

Разрабатывается компанией Microsoft с 2009 года для ее поискового сервиса Bing. Обрабатывает более 40 языков на основе статистических методов.

Яндекс.Переводчик (Россия)

В марте 2011 года российская поисковая система Яндекс запустила сервис «Яндекс.Переводчик», действующий на основе статистических методов. Он позволял переводить тексты и веб-страницы с русского на английский или украинский язык и обратно. Сейчас этот сервис предоставляет перевод для 67 языков.

Переводчик на основе ABBYY Compreno (Россия)

В 2011 году ведущая российская компания ABBYY получила от Сколково грант для создания новой технологии синтаксического и семантического анализа текста под названием ABBYY Compreno. Ядром этой технологии стала универсальная иерархия понятий и сеть отношений между ними. В идеале эта иерархия будет содержать все смыслы, о которых говорят и думают люди, с указанием связей между этими смыслами. В результате из каждого предложения можно будет выделить закодированный в нем смысл.

Разработчики планируют использовать ABBYY Compreno для решения различных задач прикладной лингвистики. На ее основе уже реализован интеллектуальный поиск, то есть поиск по смыслу, а не по ключевым словам. Сейчас ABBYY активно работает над созданием нового переводчика, который пока не получил официального названия.

Автор благодарит сотрудников компании PROMT Н.В Железняк, А.П. Молчанова, А. В. Бутакова и Т. Н. Пилатову за помощь и примеры, использованные при написании этой статьи.

Литература

Беляева Л. Н., Откупщикова М. И. Автоматический (машинный) перевод // Прикладное языкознание / под. ред. Герда А. С. СПб., 1996.

Молчанов А. Статистические и гибридные методы перевода в технологиях компании ПРОМТ. CONTROL ENGINEERING Россия #4 (46), М., 2013.

Koehn P. Statistical Machine Translation. Cambridge, UK, 2010.

Koehn P. Statistical Machine Translation System: User Manual and Code Guide. Edinburgh, UK, 2015.

Koehn, P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N., Cowan B., Shen W, Moran C., Zens R., Dyer C., Bojar O., Constantin A., and Herbst E. Moses: Open source toolkit for statistical machine translation. ACL, Prague, Czech Republic, 2007.

Manning Christopher D., Schütze Hinrich. Foundations of Statistical Natural Language Processing. MIT Press. Cambridge, MA, 1999.

Papineni K., Roukos S., Ward T., and Zhu W. BLEU: a Method for Automatic Evaluation of Machine Translation. ACL, Philadelphia, US, 2002.

Tiedemann J., Nygaard L. The OPUS corpus - parallel & free. LREC, Lisbon, Portugal, 2004

Varga D., Nemeth L., Halacsy P., Kornai A., Tron V., Nagy V. Parallel corpora for medium density languages. RANLP, Borovets, Bulgaria, 2005.

Электронные ресурсы

Moses, полный набор для создания SMT http: Apertium //www.statmt.org/moses/
LF Aligner, выравнивание текстов http://aligner.sourceforge.net/
Hunalign, алгоритм выравнивания текстов http://www.language-archives.org/item/oai:lindat.mff.cuni.cz:11372/LRT-1203,
Руководство пользователя алгоритма hunalign: http://mokk.bme.hu/resources/hunalign/
GIZA++, выравнивание по словам http://www.fjoch.com/GIZA++.html
OPUS, открытый корпус параллельных текстов http://opus.lingfil.uu.se/
Europarl, параллельные протоколы Европарламента http://www.statmt.org/europarl/
Acquis Communautaire, параллельные документы Европарламента://langtech.jrc.it/JRCAcquis.html
SRILM, создание моделей языка http://www.speech.sri.com/projects/srilm/
IRST LM, создание моделей языка http://sourceforge.net/projects/irstlm/
BLEU, метрика оценки качества http://www.nist.gov/speech/tests/mt/scoring/
METEOR, метрика оценки качества is available at http://www.cs.cmu.edu/ alavie/METEOR/
Asiya, оценка статистического перевода http://asiya.lsi.upc.edu/demo/asiya_online.php
Система Apertium, которую можно настраивать на свой язык http://www.apertium.org/index
Библиотека по машинному переводу http://www.mt-archive.info/
Лекции Михаэля Коллинза на Coursera https://www.coursera.org/course/nlangp