П

УДК 004.912

Я. С. Пикалёв
Государственное учреждение «Институт проблем искусственного интеллекта», г. Донецк
83048, г. Донецк, ул. Артема, 118-б

РАЗРАБОТКА СИСТЕМЫ НОРМАЛИЗАЦИИ ТЕКСТОВЫХ КОРПУСОВ

Ya. S. Pikalyov
Public institution «Institute of Problems of Artificial intelligence», c. Donetsk
83048, Donetsk, Artema str., 118-b

THE DEVELOPMENT OF A TEXT CORPORA NORMALIZATION SYSTEM

Данная работа посвящена задаче разработки системы для нормализации текстовых корпусов, применяющихся для обучения языковых моделей. Автором предложен гибридный метод нормализации текстовых корпусов, который был разработан на основе процедурно-декларативного подхода, а также с применением машинного обучения. Обучена нейронная сеть для синтаксического анализа входного текста. Представлен метод обработки цифробуквенных комплексов и аббревиатур. Разработаны алгоритмы «ёфикации» и «йфикации» слов. Проведены численные исследования для сравнения использования ненормализованного и нормализованного текстового корпуса для задач языкового моделирования и распознавания речи.
Ключевые слова: корпуса текстов, синтаксический анализ, автоматическое распознавание речи, машинное обучение, обработка естественного языка.

This work is devoted to the task of developing a system for the normalization of text corpora used for teaching language models. The author proposed a hybrid method of normalization of text corpora, which was developed on the basis of a procedural-declarative approach, as well as using machine learning. A neural network has been trained to parse the input text. A method for processing alphanumeric complexes and abbreviations is presented. Algorithms have been developed. "efications" and "yofications" of words. Numerical studies have been carried out to compare the use of non-normalized and normalized text corpus for the tasks of language modeling and speech recognition.
Key words: text corpora, syntactic analysis, automatic speech recognition, machine learning, natural language processing.

Введение

В общем случае задача нормализации текстов представляет собой приведение слов и выражений (лексем) естественного языка к единой, общепринятой форме. В качестве выражения могут выступать устойчивые числовые, словесные или числословесные выражения, например, условные записи дат, телефонных номеров, аббревиатуры. Под нормализацией текста в рамках данной работы понимается процесс трансформации исходного текста посредством удаления неконтекстных символов, т.е. без потери смысла исходного текста, а также преобразования символов, вносящих «шум» для понимания смысла (цифробуквенные комплексы, сокращения и т.п.).

На сегодняшний день количество нормализованных текстовых корпусов лимитировано. К тому же существующие текстовые корпуса, как правило, не нормализованы, т.к. они получены преимущественно из новостных лент; сообщений между пользователями известных соцсетей; субтитров, хранящихся на интернет-ресурсах; текстовых расшифровках выступлений, радиопередач и т.п. Использование данных корпусов без предварительной нормализации для обучения языковых моделей приводит к ухудшению качества языковых моделей, что в свою очередь приводит к ошибкам распознавания речи, т.к. на итоговую оценку влияет качество языковой модели:

$$score = (1 - k) \cdot score_{AM} + k \cdot score_{LM}, \quad (1)$$

где $score_{LM}$ – оценка гипотез из языковой модели, $score_{AM}$ – оценка гипотез из акустической модели; k – интерполяционный вес.

Данная работа ориентирована на создание автоматической системы нормализации текстовых корпусов с использованием лингвистических знаний совместно с глубоким обучением, без необходимости в дополнительном языковом модуле. Эта процедура позволит формировать нормализованные текстовые корпусы, использование которых, в отличие от ненормализованных, позволит повысить качество языковых моделей и общее качество автоматической системы распознавания речи. Помимо задачи автоматического распознавания речи, данная система подготовки корпусов может применяться для ряда задач, связанных с естественной обработкой языка.

О проблемах нормализации текстовых корпусов для задачи автоматического распознавания речи

Современные системы распознавания речи работают на основе сложнейших математических моделей, но некоторые ученые считают, что возможности применения математики для распознавания речи ограничены. Поэтому учёными разрабатываются лингвистические подходы к распознаванию, но до сих пор не было создано алгоритма более эффективного, чем математический, который позволял бы работать с языком. Лингвистическая модель в чистом виде не способна справиться с задачей. Правила лингвистики могут лишь использоваться для подкрепления математических алгоритмов.

Необходимо дополнительно рассматривать семантические ошибки (ошибки в выражениях, вследствие которых неправильно распознаётся смысл фразы, зачастую связаны с омонимами, омофонами, омографами и омоформами). Например, ненормализованный текст «этот *плот* был сочный и ароматный» должен быть преобразован в «этот *плод* был сочный и ароматный».

Другим языкам, которые исследователи именуют «*low resource languages*» [1] (в их число входит русский язык) учёные уделяют гораздо меньше внимания. Главной причиной, как можно увидеть из термина, является недостаточное количество аннотированных

речевых баз. Отдельной проблемой также является недостаточное количество нормализованных текстов, которую учёные называют «*data sparsity problem*» [2] (проблема отсутствия достаточного количества текстовых данных для статистического моделирования языка), также является недостаточное количество нормализованных текстов. Для русского языка в силу его флективности и свободного порядка слов в предложении *data sparsity problem* имеет большее влияние, чем для остальных *low-resource languages*.

Дополнительно задача осложняется наличием в русскоязычных текстах аббревиатур и вставок на латинице; неконтекстных строк и символов; различного рода ошибок и т.п.

Описание системы нормализации текстовых корпусов

Система нормализации текста состоит из нескольких модулей (рис. 1), которые делятся на 2 блока: блок предварительной обработки (*init-normalize*) и основной обработки (*core-normalize*).

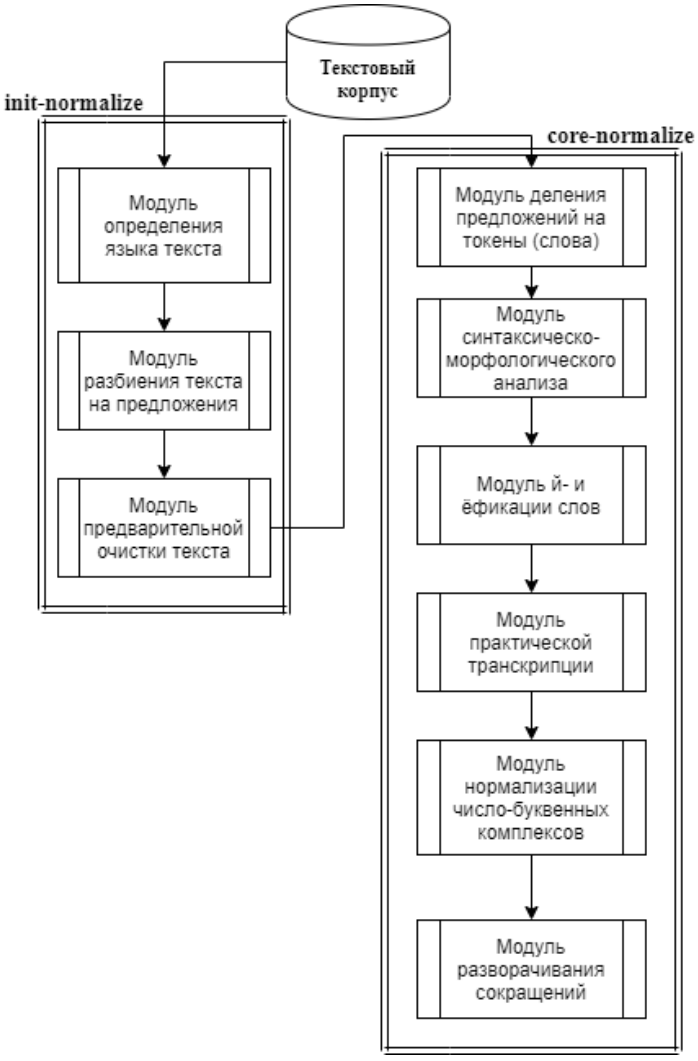

Рисунок 1 – Схема блока нормализации текста

Основные шаги алгоритма *init-normalize* состоят в следующем.

1. Считывается массив текстов $T=\{t_i\}$, где i – общее количество текстов.
2. Для каждого t_i проводится проверка на принадлежность к русскому языку, для этого используется предобученная нейросетевая модель для классификации, описанная в работе [3]. В качестве входного значения используется текст с количеством символов $l_{min}=\max(l, 100)$, где l – общее количество символов во входном тексте. В случае, если язык текста отличен от русского – используется следующий t_i. Иначе – переходим на следующий шаг.
3. Проверяется текст с количеством символов l_{min} на наличие букв «ё» и «й». Если нет ни единого символа «ё» устанавливаем значение *bool_jo*=False, нет ни единого «й» – *bool_j*=False.
4. t_i разбивается по символу окончания строки. Формируется множество $N=\{n_j\}$, где j – количество полученных абзацев.
5. n_j разбивается на предложения при помощи модуля разделения текста на предложения, использующего модель, обученную на русских текстах (дампы Wikipedia), что позволяет точно определять границы предложения (например, игнорируя знаки «.» перед сокращениями, как «г.», «в.»). Формируется множество $S=\{s_k\}$, где k – общее количество предложений.
6. Для s_k:
    a. производится очистка от тэгов, «неконтекстных символов», нестроковых значений, удаление текстовых последовательностей внутри «[]» и «()», удаление более 4 повторяющихся символов из слов, удаление предложений с верхним регистром и т.п., используя регулярные выражения для удаления или замены;
    b. удаляются строки интернет-ссылок, электронных почт, слова, содержащие символ «#», а также слова, не несущие контекстной информации;
    c. удаляются строки, в конце которых отсутствуют знаки, а также длина которых составляет менее 7 символов;
    d. производится замена римских цифр на арабские.
7. Производится корректировка строк с неверными наращиваниями (200летний, 2005г., $3), используя регулярные выражения.
8. Используя модуль исправления текстовых опечаток, описанный в работе [4], производится коррекция текстовых строк.
9. Полученные строки записываются в файл формата csv. Дополнительно вносится информация о языке текста, наличия символа «ё» и «й».

После прохода *init-normalize* используем алгоритм *core-normalize*, который состоит в следующем.

1. Считывается информация из файла с метаданными. Считываем массив предложений. Если нет букв «ё» – используется модуль ёфикации, если нет букв «й» – используется модуль йфикации.
2. Предложение делится на массив слов, используя набор правил, реализованных при помощи регулярных выражений, исключая указатели деления («+», «-», «/», а также символы пунктуации).
3. Извлекается синтаксическо-морфологическая информация.
4. Используется модуль расшифровки сокращений.
5. Используется регулярное выражение для определения, является ли слово английским или русским, или содержит численную часть строки. Для английских слов производится практическая транскрипция на рус-

ский текст, используя алгоритм, описанный в работе [3]. Если содержит численную часть строки, то используется функция согласования чисел, полученная в результате применения алгоритма синтаксического анализа.

6. Если слово полностью состоит из букв верхнего регистра – проводится проверка на количество слогов, если содержит один слог – каждая буква нормализуется в соответствии со словарём («АНБ» - «а эн бэ»), иначе слово приводится к нижнему регистру.
7. Корректируются слова, содержащие знаки препинания в конце, а также состоящие из одних знаков препинаний. Знаки препинания удаляются.
8. Слова записываются через пробел, отделяя знаком в конце строки, когда достигнут конец предложения. Полученный текст записывается в текстовый файл.

Для корректной расшифровки цифро-буквенных комплексов необходимо провести согласование чисел методами синтаксического анализа.

Разработка системы синтаксического анализа

Для задачи нормализации текстов разработан синтаксический анализатор (парсер) с применением глубоких нейронных сетей.

В качестве основных метрик для парсера используются оценка немаркированной принадлежности (*Unlabelled Attachment Score, UAS*) и оценка маркированной принадлежности (*Labelled Attachment Score, LAS*). UAS – это процент слов, чья синтаксическая позиция правильно определена; LAS – это процент слов, чья синтаксическая позиция правильно определена с помощью соответствующей метки зависимостей (*subject*, *object*) [5], [6]. UAS и LAS используют как на уровне слов (UAS_w), так и на уровне предложений (UAS_S).

При анализе доступных решений были найдены готовые модели для русского языка (по состоянию на 2018 г.): *Parsey Universal* или *SyntaxNet* (UAS_S: 81,75%; LAS_S: 77,71%) [7]; UDPipe (UAS_S: 89,8%; LAS_S: 87,2%) [8]. Но они обладают недостаточной точностью, используют медленный алгоритм парсера, кроме того, для этих моделей нельзя применить ускорение на уровне графических микропроцессоров.

Для применения ANN-метода необходима большая база обучающих примеров. При обучении парсера в настоящее время используют данные стандарта *Universal Dependencies treebank* (UDT) [9]. В Сети были найдены следующие корпуса для обучения парсера русского языка формата UDT: *Russian GSD*, *Parallel UDT*, *SynTagRus UDT*, *Taiga UDT* [10].

ANN-модель, использующаяся для разработки данного синтаксического анализатора основана на свёрточных слоях, т.к. синтаксический анализатор будет использоваться для задач нормализации текста, а данная нейросетевая архитектура является оптимальной для этой цели [11]. В связи с этим использована нейронная сеть, состоящая из 3-х свёрточных ANN, с использованием слоя внимания (*attention*) для формирования контекстных векторов – 3-CNN-attention.

Механизмы внимания могут быть использованы для повышения производительности нейронных сетей. В модели со вниманием (рис. 2) вместо построения одного контекстного вектора из последнего скрытого состояния сети создаётся контекстный вектор для каждого входного слова. То есть если в исходном документе N уникальных слов, то должно быть создано N контекстных векторов, а не один.

Преимущество применения данного подхода состоит в том, что закодированная информация хорошо декодируется моделью.

Формула контекстного вектора для модели внимания имеет вид:

$$c_t = \sum_{j=1}^{T_x} a_{tj} h_j \,, \tag{2}$$

где a_{tj} – веса для каждого скрытого состояния h_j (оценка внимания):

$$a_{tj} = \frac{\exp(e_{tj})}{\sum_{T_x}^{k=1} \exp(e_{tk})} \,, \tag{3}$$

где e_{tj} – модель вложения документа:

$$e_{tj} = a(s_{t-1}, h_j) \,, \tag{4}$$

где s_t – скрытое состояние сети:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \,. \tag{5}$$

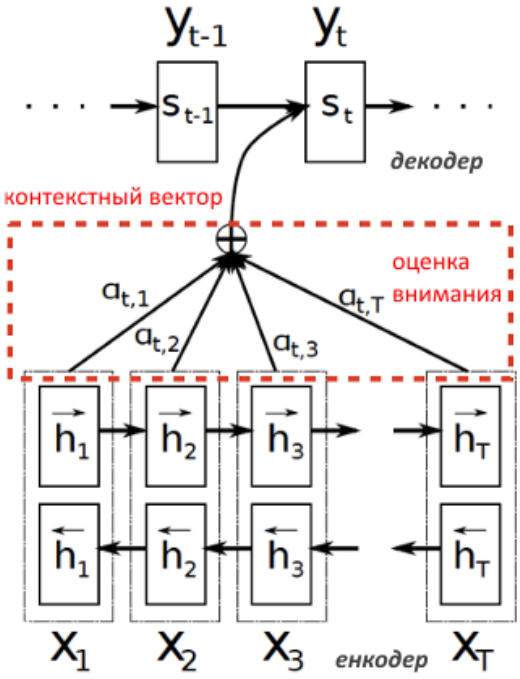

Рисунок 2 – Архитектура модели со вниманием

Архитектура используемой для парсера сети изображена на рис. 2. Каждый свёрточный слой содержит 200 слоёв в глубину, а данные, подающиеся на вход нейронной сети, представлены вектором размерностью в 64 единиц (200×64 признаков).

Главная идея данной нейронной сети заключается в формировании контекстного вектора для каждого слова в предложении. Слой сверточных сетей (3-CNN, рис. 3) выполняет как роль парсера, так и морфоанализатора (определяет часть речи и лемму).

На рис. 4 отображены результаты обучения ANN-модели синтаксическо-морфологического анализатора по критериям LAS и UAS (на уровне предложений).

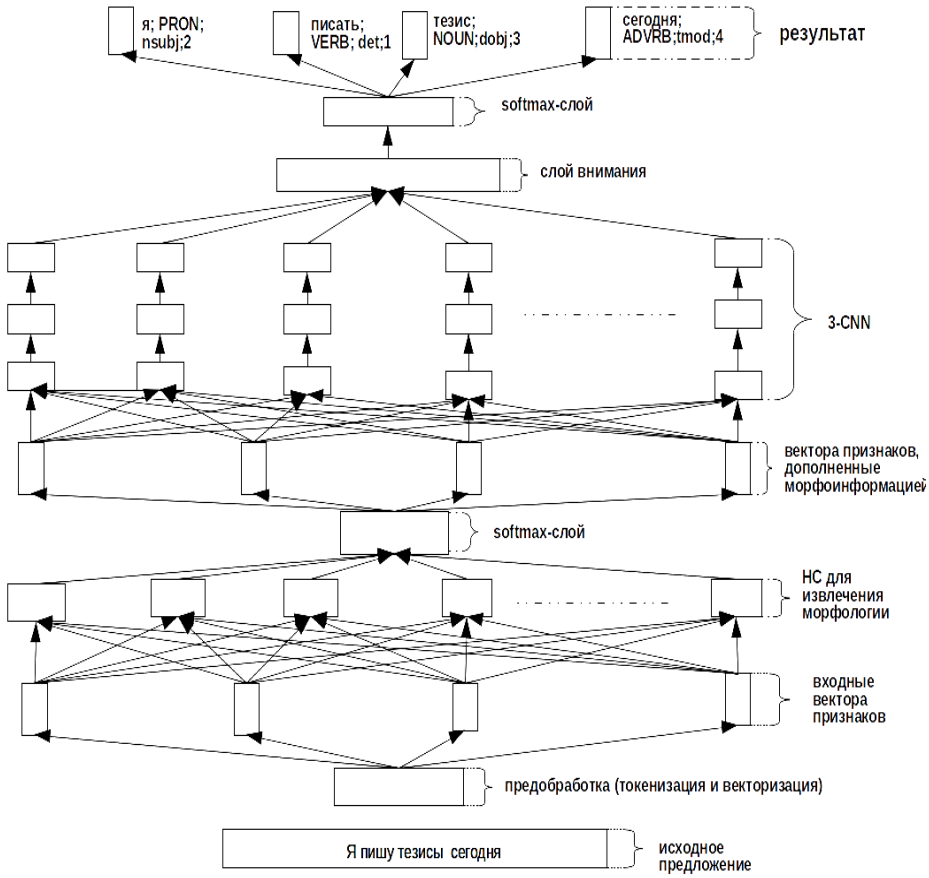

Рисунок 3 – Нейросетевая архитектура системы синтаксического анализа

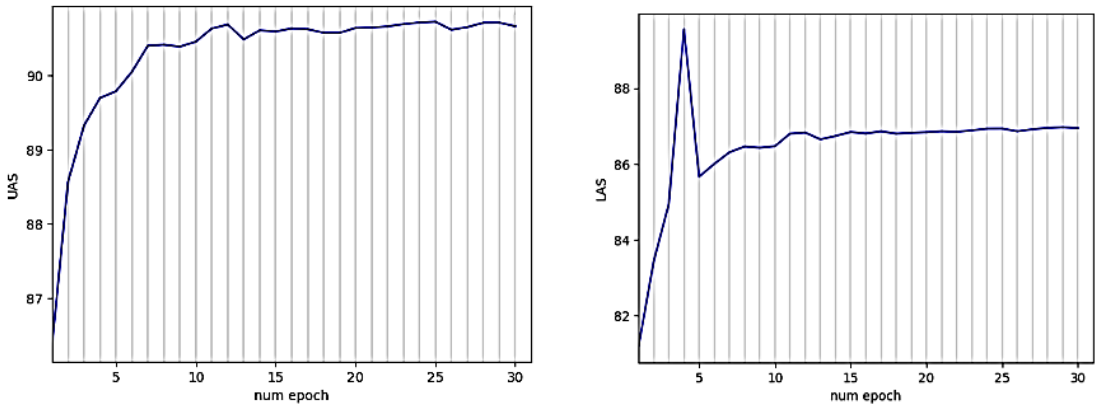

Рисунок 4 – Зависимость величины критериев UAS (слева) и LAS (справа) от количества эпох обучения системы синтаксического анализа

Одновременное использование как синтаксической, так и морфологической информации в качестве входных данных позволяет добиться улучшения в точности. В качестве входной информации используются слова с соответствующей морфологической информацией (слово: мам; лемма: мама; часть речи: NOUN), а в качестве соответствующих меток (классов) – синтаксическая позиция и метка (позиция: 1; метка: nsubj).

Предложенная модель синтаксического анализатора дает качество по критерию UAS – 94,3%, по критерию LAS – 90,2%.

Разработка метода обработки цифробуквенных комплексов и аббревиатур

Модуль согласования чисел предназначен для автоматической нормализации цифробуквенных комплексов и аббревиатур (трансформации в буквенные аналоги). Алгоритм модуля согласования чисел состоит в следующем.

1. В предложении ищется слово, состоящее из числа или имеющее численную часть.
2. В случае, если такое вхождение слово найдено – определяется шаблон слова:
   - шаблон наращивания («10-й», «10й», «10–летний», «10летний»);
   - шаблон аббревиатуры и цифры («А1Б», «ТУ–104», «104–ТУ»);
   - шаблон времени и счёта («12:30», «12–30», «12:30:30»);
   - шаблон дат («01.05.2018», «01/05/2018», «2018–05–01»);
   - шаблон телефонных номеров;
   - шаблон дробей и адресов («3/4»);
   - шаблон чисел с плавающей точкой («0.5», «0,5»);
   - шаблон простого числа, который дополнительно подразделяется на «количественное числительное» + «сокращение» («5 кг»); «порядковое числительное» + «сокращение» («5-ый кг»); «количественное числительное» («5»).

Стоит отметить, что для определения типа шаблона простого числа использовались словари с наиболее известными сочетаниями пар. Данную задачу можно решить с использованием sequence-to-sequence (seq2seq) моделей [12], но этот подход усложнит вычислительный процесс. Подход с использованием словарей позволяет определить шаблон для большинства известных случаев.

3. В соответствии с шаблоном происходит нормализация текста:
   - для шаблона наращивания используется таблица окончаний, в соответствии с которой склоняется слово; в случае «10-летний» – численная часть приводится в дательный падеж «десятилетний»;
   - для шаблона аббревиатур цифровая и буквенная части разделяются «А1Б» – «А», «1», «Б». Затем буквы соотносятся со словарём («А1Б» – «а один бэ»);
   - для шаблона времени и счёта используется система синтаксического анализа. Для этого используются 2 синтаксически связанных слова с вхождением шаблона (ведётся поиск предлога и самостоятельных частей речи). В случае предлога используем словарь, для остальных – извлекаем падеж из самостоятельной речи, склоняем оба числа. Стоит отметить, что существуют так называемые «двойные падежи», которые отличаются от стандартных шаблонов формирования падежей, эта проблема частично решается составлением словаря или набора правил для проблемных сокращений (как правило, в качестве определителя падежа используются предлоги);
   - нормализация шаблона дат проводится аналогично шаблону времени и счёта;
   - при нормализации шаблона телефонных номеров учитывается два типа этого шаблона:
     a) шаблон для телефонных номеров, записанных в виде «361-72-72». Для этого шаблона разделяются числа, используя разделитель «–». Полученные числительные приводятся в текстовый вид как начальные формы количественных числительных;
     b) шаблон телефонных номеров, записанных в виде «+380714648734». Последние 9 цифр разделяются как «714», «64», «87», «34». Все остальные символы разделяются посимвольно:

– нормализация шаблона дробей и адресов производится аналогично шаблону времени и счёта, но изменяется лишь второе число; для адресов (просматривается, начинается ли предыдущее слово с большой буквы) – преобразуем каждое число в количественное числительное;
– для обработки шаблона чисел с плавающей точкой делим числа по разделителю и смотрим на длину второго числа (удаляя нули, если они находятся перед цифрами: «0.005»), в соответствии с длиной используем словарь и добавляем слово в конец, дополнительно используем подход из шаблона аббревиатуры и счёта для добавленного слова в конец;
– для шаблона простого числа ищем синтаксически связанные слова, если из списка сокращений слово находится на расстоянии 0...2 – по сокращению определяем вид числительного; далее ищем соседние синтаксические слова (местоимение, предлог, имя прилагательное). Число переводим в строковую форму, а сокращение в полную форму, изменяя его, используя морфологическую информацию соседнего синтаксического слова («в 5 в.» – «в пятом веке»).

В русскоязычных текстах, как правило, буква «ё» заменяется на «е», благодаря чему упрощается и ускоряется распознавание слов в ходе морфологического анализа для большинства текстов. Для построения качественной языковой модели необходимо провести обратную операцию – заменить «е» на «ё» в соответствии с правилами русского языка там, где «ё» должна употребляться.

Еще одной проблемой при обработке русскоязычного текста является наличие опечаток, где «й» заменяется на «и». В связи с этим для проведения нормализации необходимо разработать и реализовать алгоритмы «ёфикации» и «йфикации» слов.

Разработка алгоритмов «ёфикации» и «йфикации» слов

Алгоритмы «ё-» и «йфикации» предназначены для автоматического исправления ошибок в словах, а именно неверного использования символов «е» (или «и») и «ё» (или «й»). Эти модули основаны на 2-х типах словарей, сформированных автором:

1) словарь, содержащий парадигмы слов с символами «ё» («й»).
2) словарь, содержащий парадигмы слов, в которых в одной и той же позиции может использоваться как символ «ё» («й»), так и символ «е» («и»): «небо» – «нёбо»; «мои» – «мой».

Принцип работы *методов «ёфикации» и «йфикации»* следующий.

1. Инициализируется слово.
2. Производится проверка на наличие в слове символов «ё» и «е» (или «й» и «и»). Если данные символы присутствуют, то переходим к следующему шагу. Иначе возвращаем исходное слово.
3. В слове символы «ё» (или «й») заменяются на символы «е» (или «и»). Производится поиск слова в исходном словаре, который содержит парадигмы слов с символами «ё» (или «й»), и извлекается его найденное значение в случае, если слово есть в данном словаре («елка»: «ёлка» или «иод»: «йод»; данный словарь состоит из пар слов). Иначе переходим к следующему шагу.
4. Слово ищется в словаре, который содержит парадигмы слов, в которых в одной и той же позиции может использоваться как символ «ё» (или «й»), так и символ «е» (или «и»), если такое вхождение слова есть, то извлекается его найденное значение. Иначе переходим к следующему шагу.

5. Если слова нет ни в одном из словарей, то по умолчанию символы «ё» (или «й») заменяются на «е» (или «и»), т.е. используется «ефицированное» (или «ифицированное») значение слова, т.к. это слово с большой вероятностью относится к словам, содержащих символ «е» (или «и»). Существует вероятность того, что это слово внесловарное («Тёёлё» или «днровскии»), но данную проблему практически невозможно решить ни применением шаблонов, ни применением машинного обучения (т.к. большинство текстов «ефицировано» или «ифицировано»). Данная проблема решается лишь за счёт расширения существующих словарей, поэтому исходное слово дополнительно выводится в отдельный текстовый файл для дальнейшего рассмотрения.

Для метода «йфикации» дополнительно создан алгоритм с учётом контекста в предложении.

Работа *метода «йфикации» с учётом контекста в предложении* состоит в следующем.

1. Взять исходное слово, а также предложение, где встречается данное слово.
2. Проверить слово на наличие символов «и» или «й». Если данные символы есть – переходим к следующему шагу. Иначе возвращаем исходное слово.
3. Использовать систему синтаксического анализа для данного предложения, т.е. построить синтаксическое дерево связей.
4. Используя синтаксическое дерево связей, взять 4 синтаксически близких слова с искомым (2 слева и 2 справа).
5. Так как система синтаксического анализа позволяет извлекать информацию только о части речи, то дополнительно используется модель морфологического анализа, RNNMorph [13], позволяющая извлечь расширенную морфологическую информацию для каждого слова в исходном предложении.
6. Найти ближайшее слово, являющееся прилагательным, местоимением, существительным, глаголом (т.е. содержащее информацию о числе). Если таковое слово найдено – переходим к следующему шагу, иначе используем алгоритм «йфикации» без учёта контекста.
7. Если ближайшее слово имеет множественное число, то и «йфицированное» слово имеет множественное число (по аналогии определяется единственное число).
8. Разбиваем слово на 3 части, используя алгоритм стемматизации [14]: приставку, корень и суффикс с окончанием.
9. Применяем запрограммированные правила «йфикации» к корню; производится проверка на наличие символов в части с окончанием, если присутствуют символы, то производится проверка по морфологической информации, а именно числу, «йфицированного» слова; если число множественное – в данном слове будет использоваться символ «и»; единственное – символ «й». Если часть с окончанием пуста, то далее используется алгоритм «йфикации» без учёта контекста.

Стоит отметить неточности в правилах орфографии относительно написания «й», а именно то, что после согласных буква «*й*» не пишется» [15]. Это не соответствует действительности, примером чему могут служить слова «безйотовый», «безйодовый». Данная погрешность устранена в [16].

Численные исследования

Для оценки качества ЯМ используют коэффициент неопределенности (перплексия, *perplexity*). Для тестовых данных T, состоящих из предложений $(t_1, t_2, \dots, t_{l_T})$, содержащих суммарно WT слов, вероятность определяется как произведение вероятностей для каждого из предложений:

$$P(t) = \prod_{k=1}^{l_t} P(t_k). \quad (6)$$

Перплексия (PPL) определяется следующим образом:

$$PPL(T) = 2^{H(T)} = P(T)^{\frac{1}{W_T}}. \quad (7)$$

Чем меньше эта величина, тем лучше модель предсказывает появление слов в документах текстового корпуса. Между перплексией и количеством неправильно распознанных слов существует сильная корреляция [17]: чем меньше взаимная энтропия и перплексия, тем лучше модель соответствует тестовым данным.

Для оценки результатов распознавания речи использовалась метрика WER – отношение количества слов с неверным ударением к общему количеству слов (*Word Error Rate*).

В качестве обучающего материала для формирования АМ использованы два речевых корпуса – *VoxForge* [18]; общей продолжительностью около 20 часов, из которых предварительно удалены неконтекстные речевые данные.

В качестве основной акустической модели была использована трифонная акустическая модель с количеством состояний для скрытой марковской модели равным 2 500 и количеством гауссиан – 20 000. В качестве языковой модели была использована триграмная языковая модель. Обучающая и тестовая выборки с аудиоданными разделены в отношении 95/5.

Текстовый корпус был составлен из данных, извлеченных из Сети, и составляет 28,8 млн слов, 2,2 млн предложений. Ненормализованный (*Raw*) и нормализованный (*Normalize*) текстовые корпуса описаны в табл. 1.

Результаты численных экспериментов описаны в табл. 2.

Таблица 1 – Описание текстовых корпусов

	Кол-во униграм	Кол-во биграм	Кол-во триграм
Raw	132 тыс.	2,1 млн	1,1 млн
Normalize	136 тыс.	2,5 млн	1,5 млн

Таблица 2 – Результаты численных исследований

	PPL	WER, %
Raw	652,43	37,6
Normalize	546,32	33,1

Выводы

Предложенный подход для создания автоматической системы формирования нормализованного текстового корпуса позволяет улучшить результат языкового моделирования на 106.11 по метрике перплексии; для распознавания речи – на 4,5 %. Этот результат обеспечивается с учетом лингвистических норм русского языка, а также при помощи глубокого обучения.

Предложенная система получения нормализованных текстовых корпусов может быть использована не только для задач языкового моделирования и распознавания речи, но и в ряде задач, связанных с естественной обработкой языка.

Список литературы

1. THE HOLY GRAIL: AUTOMATIC SPEECH RECOGNITION FOR LOW–RESOURCE LANGUAGES. [Электронный ресурс]. – Режим доступа: https://beckman.illinois.edu/news/2016/03/asr–jyothi (Дата обращение: 20.05.2017).
2. Allison, B. Another Look at the Data Sparsity Problem [Текст] / B. Allison, D. Guthrie, L. Guthrie // Text, Speech and Dialogue, Lecture Notes in Computer Science. – 2006. – Vol. 4188. – P. 327–334.
3. Пикалёв Я. С. Система автоматической генерации транскрипций русскоязычных слов-исключений на основе глубокого обучения [Текст] / Я. С. Пикалёв, Т. В. Ермоленко // Проблемы искусственного интеллекта. – 2019. – № 4 (15). – С. 35–51.
4. Ермоленко Т. В. Классификация ошибок в тексте на основе глубокого обучения [Электронный ресурс] / Ермоленко Т. В. // Проблемы искусственного интеллекта. – 2019. – №3 (14). – URL: https://cyberleninka.ru/article/n/klassifikatsiya-oshibok-v-tekste-na-osnove-glubokogo-obucheniya (дата обращения: 05.08.2022).
5. Jurafsky D. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (University of Colorado, Boulder) Upper Saddle River, NJ: Prentice Hall (Prentice) [Текст] / D. Jurafsky, J.H. Martin // Computational Linguistics. – 2000. – Т. 26, № 4.
6. Carroll J. Dependency Parsing Sandra Kübler, Ryan McDonald, and Joakim Nivre (Indiana University, Google Research, and Uppsala and Växjö Universities) Morgan & Claypool (Synthesis Lectures on Human Language Technologies, edited by Graeme Hirst, volume 2), 2009, xii+ [Текст] / J. Carroll // Computational Linguistics. – 2010. – Т. 36, № 1.
7. Syntaxnet Parsey McParseface wrapper for POS tagging and dependency parsing [Электронный ресурс]. – URL: https://github.com/spoddutur/syntaxnet (дата обращения: 20.09.2018).
8. UDPipe Models [Электронный ресурс]. – URL: http://ufal.mff.cuni.cz/udpipe/models (дата обращения: 20.09.2018).
9. Universal Dependencies [Электронный ресурс]. – URL: https://universaldependencies.org/ (дата обращения: 25.09.2018).
10. Comparison of Treebank Statistics [Электронный ресурс]. – URL: https://universaldependencies.org/treebanks/ru-comparison.html (дата обращения: 20.09.2018).
11. Пикалёв Я.С. Разработка синтаксического анализатора русского языка на основе глубоких нейронных сетей [Текст] / Я.С. Пикалёв // Донецкие чтения 2018: образование, наука, инновации, культура и вызовы современности: Материалы III Международной научной конференции (Донецк, 25 октября 2018 г.). – Том 1: Физико-математические и технические науки/ под общей редакцией проф. С. В. Беспаловой. – Донецк : ДонНУ, 2018. – С. 240–243.
12. Lourentzou I. Adapting sequence to sequence models for text normalization in social media [Текст] / I. Lourentzou, K. Manghnani, C.X. Zhai // Proceedings of the 13th International Conference on Web and Social Media, ICWSM 2019. – 2019.
13. Anastasyev D.G. Improving part-of-speech tagging via multi-task learning and character-level word representations [Текст] / D. G. Anastasyev, I. O. Gusev, E. M. Indenbom // Komp'juternaja Lingvistika i Intellektual'nye Tehnologii. – 2018. – Тт. 2018-May.
14. Porter M. F. Snowball: A language for stemming algorithms [Электронный ресурс] / Porter M. F. – URL: https://pdfs.semanticscholar.org/0d8f/907bb0180912d1e1df279739e45dff6853ee.pdf (дата обращения: 20.05.2019).
15. Скобликова Е.С. Правила русской орфографии и пунктуации. Полный академический справочник [Текст] / Е.С. Скобликова; ред. В.В. Лопатин. – М.: Эксмо, 2006. – 480 с.
16. Бешенкова Е. В. Объяснительный русский орфографический словарь-справочник [Текст] / Е. В. Бешенкова, Л. К. Чельцова, О. Е. Иванова; ред. Е. В. Бешенкова. – М. : АСТ-Пресс, 2018. – 592 с.
17. Chen S. Evaluation metrics for language models [Текст] / S. Chen, D. Beeferman, R. Rosenfeld // Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop. – 1998.
18. Шмырёв Н. В. Свободные речевые базы данных voxforge.org [Текст] / Н. В. Шмырёв // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог» (Бекасово, 4–8 июня 2008 г.). – 2008. – № 7(14). – С. 585–588.

References

1. *THE HOLY GRAIL: AUTOMATIC SPEECH RECOGNITION FOR LANGUAGES WITH LOW RESOURCE*. [Electronic resource]. Access mode: https://beckman.illinois.edu/news/2016/03/asr-jyothi (Publication date: 05/20/2017).
2. Allison, B. D. Guthrie, L. Guthrie Another Look at the Data Sparsity Problem. *Text, Speech and Dialogue, Lecture Notes in Computer Science*, 2006, Vol. 4188, P. 327–334.
3. Pikaliov Ya.S., Ermolenko T.V. Sistema avtomaticheskoy generatsii transkriptsiy russkoyazychnykh slov-isklyucheniy na osnove glubokogo obucheniya [System of automatic generation of transcription of Russian-language exception words based on deep learning]. *Problemy iskusstvennogo intellekta* [Problems of artificial intelligence], 2019, Vol. 4, No. 15, pp. 35-51.
4. Ermolenko T.V. Klassifikatsiya oshibok v tekste na osnove glubokogo obucheniya [Classification of errors in the text based on deep learning] *Problemy iskusstvennogo intellekta* [Problems of artificial intelligence], 2019, No.3 (14), URL: https://cyberleninka.ru/article/n/klassifikatsiya-oshibok-v-tekste-na-osnove-glubokogo-obucheniya (accessed: 05.08.2022).
5. Jurafsky D., Martin J.H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (University of Colorado, Boulder) Upper Saddle River, NJ: Prentice Hall (Prentice). *Computational Linguistics*. 2000. T. 26. № 4.
6. Carroll J. Dependency Parsing Sandra Kübler, Ryan McDonald, and Joakim Nivre (Indiana University, Google Research, and Uppsala and Växjö Universities) Morgan & Claypool (Synthesis Lectures on Human Language Technologies, edited by Graeme Hirst, volume 2), 2009, xii+. *Computational Linguistics*. 2010. T. 36. No. 1.
7. *Syntaxnet Parsey McParseface wrapper for POS tagging and dependency parsing* [Electronic resource]. URL: https://github.com/spoddutur/syntaxnet (Publication date: 20.09.2018).
8. *UDPipe Models* [Electronic resource]. URL: http://ufal.mff.cuni.cz/udpipe/models (Publication date: 20.09.2018).
9. *Universal Dependencies* [Electronic resource]. URL: https://universaldependencies.org/ (Publication date: 25.09.2018).
10. *Comparison of Treebank Statistics* [Electronic resource]. URL: https://universaldependencies.org/treebanks/ru-comparison.html (Publication date: 20.09.2018).
11. Pikalv Ya.S. Razrabotka sintaksicheskogo analizatora russkogo yazyka na osnove glubokikh neyronnykh setey [Development of a parser of the Russian language based on deep neural networks]. *Donetskiye chteniya 2018: obrazovaniye, nauka, innovatsii, kul'tura i vyzovy sovremennosti: Materialy III Mezhdunarodnoy nauchnoy konferentsii* [Donetsk Readings 2018: Education, Science, Innovation, Culture and Challenges of Modernity: Materials of the III International Scientific Conference] (Donetsk, October 25, 2018). Volume 1: Physical, Mathematical and Technical Sciences/ edited by Prof. S. V. Bespalova. Donetsk: DonNU, 2018. pp. 240-243.
12. Lourentzou I., Manghnani K., Zhai C.X. Adapting sequence to sequence models for text normalization in social media. *Proceedings of the 13th International Conference on Web and Social Media, ICWSM 2019*. 2019.
13. Anastasyev D.G., Gusev I.O., Indenbom E.M. Improving part-of-speech tagging via multi-task learning and character-level word representations. *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*. 2018. Тт. 2018-May.
14. Porter M.F. *Snowball: A language for stemming algorithms* [Electronic resource]. URL: https://pdfs.semanticscholar.org/0d8f/907bb0180912d1e1df279739e45dff6853ee.pdf (Publication date: 20.05.2019).
15. Skoblikova E.S. *Pravila russkoy orfografii i punktuatsii. Polnyy akademicheskiy spravochnik* [Rules of Russian spelling and punctuation. The complete academic handbook] / E.S. Skoblikova; edited by V.V. Lopatin. M., Eksmo, 2006. 480 p.
16. Beshenkova E.V. *Ob"yasnitel'nyy russkiy orfograficheskiy slovar'*-spravochnik [Explanatory Russian spelling dictionary--reference book] / E.V. Beshenkova, L.K. Cheltsova, O.E. Ivanova; ed. E.V. Beshenkov. M., AST-Press, 2018. 592 p.
17. Chen S., Beeferman D., Rosenfeld R. Evaluation metrics for language models. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. 1998.
18. Shmyrev N.V. Svobodnyye rechevyye bazy dannykh voxforge.org [Free speech databases voxforge.org] *Komp'yuternaya lingvistika i intellektual'nyye tekhnologii: Po materialam yezhegodnoy Mezhdunarodnoy konferentsii «Dialog» (Bekasovo, 4–8 iyunya 2008 g.)*. [Computational linguistics and intellectual technologies: Based on the materials of the annual International Conference "Dialogue" (Bekasovo, June 4-8, 2008)]. – 2008. – № 7(14). – Pp. 585-588.

RESUME

Ya. S. Pikalyov

The Development of a Text Corpora Normalization System

The problem considered in this paper relates to one of the main tasks of natural language processing of text normalization. The development of an optimal system of normalization of the Russian-language text is one of the key problems of the direction of natural language processing (NLP). The relevance of this problem is confirmed by the widespread introduction of such systems into most types of software products (search engines, recommendation systems, etc.), as well as the use of NLP and artificial intelligence by researchers and developers for other tasks. At the moment, these systems are implemented using phonetic rules or using machine learning. In this regard, an urgent task is to develop an automatic normalization system for text corpora.

The following methods are used in this article: classification methods, text tokenization methods, the method of evaluation by the error of recognition of correspondences; the Python programming language was used for software implementation.

An automatic system of normalization of text corpora has been formed; training data sets have been collected for the task of text syntactic analysis, as well as for the task of language modeling and automatic speech recognition; a model of syntactic analysis has been obtained; methods for processing alphanumeric complexes and abbreviations have been formed; algorithms for "efication" and "yification" have been formed.

The proposed approach for creating an automatic normalization system for text corpora allowed to improve the quality of the language model by 106.11 according to the metric of perplexity, and by 4.5% according to the metric of word-by-word recognition error.

The proposed system for obtaining normalized text corpora can be used not only for the tasks of language modeling and speech recognition, but also in a number of tasks related to natural language processing.

РЕЗЮМЕ

Я. С. Пикалёв

Разработка системы нормализации текстовых корпусов

Проблема, рассмотренная в данной работе, относится к одной из основных задач естественной обработки языка нормализации текста. Разработка оптимальной системы нормализации русскоязычного текста является одной из ключевых проблем направления обработки естественного языка (*natural language processing, NLP*). Актуальность данной проблемы подтверждается широким внедрением подобных систем в большинство видов программных продуктов (поисковые системы, системы рекомендаций и т.п.), а также использованием исследователями и разработчиками для других задач NLP и искусственного интеллекта. На текущий момент данные системы реализованы при помощи фонетических правил или с применением машинного обучения. В связи с этим актуальной задачей является разработка автоматической системы нормализации текстовых корпусов.

В данной статье использованы следующие методы: методы классификации, методы токенизации текста, метод оценки по показателю ошибки распознавания соответствий; для программной реализации был использован язык программирования Python.

Сформирована автоматическая система нормализации текстовых корпусов; собраны обучающие наборы данных для задачи синтаксического анализа текста, а также для задачи языкового моделирования и автоматического распознавания речи; получена модель синтаксического анализа; сформированы методы обработки цифро-буквенных комплексов и аббревиатур; сформированы алгоритмы «ёфикации» и «йфикации».

Предложенный подход для создания автоматической системы нормализации текстовых корпусов позволил улучшить качество языковой модели на 106.11 по метрике перплексии, и на 4.5 % по метрике пословной ошибки распознавания.

Предложенная система получения нормализованных текстовых корпусов может быть использована не только для задач языкового моделирования и распознавания речи, но и в ряде задач, связанных с естественной обработкой языка.

Статья поступила в редакцию 15.04.2022.