Московский государственный университет имени
М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра системного программирования

Курсовая работа

«Определение языка сообщений социальной сети Twitter»

Выполнил студент 327 группы

М. А. Кольцов

Научный руководитель

В. Д. Майоров

Москва, 2014

Содержание

Аннотация

В данной работе рассматривается задача определения языка сообщений социальной сети Twitter: с помощью имеющейся выборки твитов нужно построить классификатор, который будет для новых текстов выдавать предполагаемый язык. Сравниваются три самостоятельных системы для решения этой задачи с тремя авторскими реализациями, основанными на актуальных статьях. Показывается конкурентоспособность имеющихся решений, а также пригодность к решению задачи одного из предложенных.

Введение

В настоящее время человечество имеет доступ к огромному запасу знаний, накопленному за тысячи лет. Немалая часть этих знаний представляется в виде текстов на различных языках. В связи с этим активно разрабатываются методы, предназначенные для автоматического извлечения и преобразования информации, данной в символьном представлении. Возникло научное направление «обработка естественных языков». Одной из его фундаментальных задач является определение языка текста.

Стандартным подходом к этой задаче является применение машинного обучения. А именно, если у нас есть корпус документов на нескольких языках, то можно «предсказать» язык поступившего на рассмотрение документа, сравнив его с имеющимися. В общем случае, нужно по имеющимся данным построить модель, а затем все действия с текстами проводить в терминах этой модели. Классическим примером является метод, описанный в 1994 году (см. [1]) каждому документу сопоставим «профиль документа» — упорядоченный по числу вхождений список n-грамм — последовательностей длины n подряд идущих символов. «Профиль языка» — это совокупность профилей документов, которые имеются на этом языке. Теперь, если нужно для какого-то документа определить язык, то последовательность действий такова:

1. Составить профиль документа
2. Сравнить полученный профиль с имеющимися профилями языков
3. Вернуть в качестве результата язык, чей профиль наиболее похож на профиль документа

Вышеописанный метод плохо работает для сообщений, содержащих ошибки, потому что соответствующие ошибкам n-граммы реже встречаются в профилях языков и не попадают в модель. В то же время, поток информации в виде коротких неформальных сообщений нельзя игнорировать — в социальной сети Twitter среднее количество сообщений в день составляет более пятидесяти восьми миллионов[1], а длина каждого ограничена 140 символами. Такой формат текстов обусловил появление новых алгоритмов. В данной работе рассматриваются современные методы решения задачи определения языка, а также предлагается улучшение для одного из них. Проводится тестирование, показывающее превосходство по качеству распознавания языка полученного результата над существующими.

[1]По данным http://www.statisticbrain.com/twitter-statistics/ на 1 января 2014 года

1 Постановка задачи

1.1 Формальное описание задачи автоматического определения языка

Пусть L - множество меток, сопоставленных естественным языкам. По заданному тренировочному корпусу

$$T = \{(msg_1, lang_1), (msg_2, lang_2), \ldots, (msg_N, lang_N)\}$$

(где $msg_i, i \in \overline{1..N}$, — текст на естественном языке, $lang_i \in L$ — метка этого языка) нужно построить классификатор, который произвольному входному сообщению new_msg на языке $some\_lang$ сопоставит метку $l \in L$, соответствующую этому языку, или же сообщит, что язык текста невозможно достоверно распознать.

1.2 Цели и задачи курсовой работы

В данной работе в качестве текстов выступают так называемые «твиты» — сообщения из социальной сети Twitter[2], а множество L соответствует 18 языкам, которые можно разделить на несколько групп по типу алфавита:

- Основанные на кириллице: болгарский, чувашский, русский, татарский, украинский
- Арабские: арабский, персидский (фарси), урду
- Латинские: нидерландский, французский, английский, немецкий, итальянский, испанский, турецкий
- Деванагари: хинди, маратхи, непальский

Также происходит предобработка твитов — *нормализация* — удаление некоторых частей сообщения так, чтобы для всего тренировочного корпуса выполнялись некие свойства, например, «во всех твитах отсутствуют гиперссылки».

Цели работы:

1. Исследовать современные решения задачи автоматического определения языка коротких сообщений

[2]https://twitter.com/

2. Провести совместное сравнительное тестирование некоторых методов решения задачи и выяснить, действительно ли они показывают конкурентноспособное качество классификации

3. Исследовать зависимость качества классификации от различных факторов: степени нормализации сообщений, мощности обучающей выборки

4. Исследовать возможность улучшения какого-либо алгоритма решения задачи автоматического определения языка коротких сообщений

В рамках выполнения цели работы необходимо решить следующие задачи:

1. Собрать обучающий корпус, состоящий не менее чем из пятисот твитов для каждого языка

2. Реализовать несколько методов решения задачи

2 Обзор существующих решений

2.1 Подход, основанный на подсчёте частоты n-грамм

Для каждого языка составляется список из K самых часто встречающихся n-грамм. Для new_msg составляется аналогичный список, для него определяется наиболее «похожий» из списков языков и на основании этого делается вывод о языке сообщения.

«Похожесть» двух упорядоченных списков характеризуется метрикой *out-of-place*: пусть в одном списке n-грамма x стоит на позиции i, а в другом — j, тогда к расстоянию между списками добавляется $|i - j|$.

Кавнар и Тренкль, авторы метода, в своей работе [1] выдвинули гипотезу, что ≈ 300 самых часто используемых n-грамм сильно зависят от языка, и экспериментально подтвердили её. Также они предлагают использовать длину n-грамм вплоть до 5.

Этим подходом пользуются многие программные продукты, о которых речь пойдёт в разделе 4.

Плюсы и минусы:

+ Небольшой размер модели
+ Высокая скорость обучения и классификации
− Качество классификации сильно зависит от длины текста [3]

2.2 Prediction by partial matching

Составляется кодирование (какое именно — зависит от варианта PPM), основываясь на количествах вхождений в документы цепочек из не более чем n символов (более коротким кодовым словам соответствуют более часто встречающиеся последовательности). Затем для new_msg вычисляется длина кода при помощи полученного кодирования и выбирается наиболее компактно кодирующий язык.

Такой подход предлагается, например, в работе [3].

Плюсы и минусы:

+ Один параметр — n — для настройки метода
+ Показывает высокое качество классификации [4]
− Нуждается в больших объёмах обучающих данных

[3] Как показали сами авторы в [1]

[4] По результатам [3]

2.3 Подходы, использующие алгоритмы машинного обучения

Из документов выделяются признаки, характеризующие их, и каждому документу сопоставляется вектор признаков. Таким образом, получается задача классификации в общем виде. Поскольку это более широкая задача, она была лучше изучена и есть множество известных подходов для её решения. Один из примеров — метод опорных векторов.

Благодаря гибкости в выборе признаков, метод является перспективным по отношению к определению языка твитов. Так, например, в работах [3] [4] используются такие признаки:

- количество вхождений n-грамм
- имя пользователя в Twitter и его местоположение
- язык интерфейса
- наиболее вероятный язык в предыдущих k твитах этого пользователя
- наиболее вероятный язык сущностей, на который ссылается пользователь в сообщении: текстов других пользователей, ссылок, тэгов.

Плюсы и минусы:

\+ Разработанный математический аппарат

\+ Возможность адаптации для выбранной предметной области

− Нужно тщательно подбирать признаки и алгоритм для получения хорошего качества классификации

2.4 Подход LIGA, основанный на графах

Метод предложен в работе [2] и состоит в следующем. По каждому документу обучающей выборки строится граф (V, E): каждой вершине $v \in V$ графа соответствует пара $(trigram, count)$, где $trigram$ — это одна из триграмм, встретившихся в тексте, а $count$ - количество её вхождений. Будем обозначать за v_{tr} соответствующую вершине v триграмму ($v_{tr} \neq u_{tr}\ \forall u \in V/\{v\}$), за v_{cnt} - количество её вхождений. Ориентированное ребро $(u, v) \in E$ соответствует тому, что в тексте u_{tr} следует непосредственно перед v_{tr}, а $(u, v)_{cnt}$ — количество таких вхождений.

Далее, выполняется объединение полученных графов для всех документов одного языка $l \in L$: если у двух графов (V', E') и (V'', E'') есть

вершины $v' \in V'$ и $v'' \in V''$ такие, что $v'_{tr} = v''_{tr}$, то в результирующем графе (V_l, E_l) должна быть вершина $v \in V_l$ такая, что

$$v_{tr} = v'_{tr} = v''_{tr}$$

$$v_{cnt} = v'_{cnt} + v''_{cnt}$$

Те вершины $v' \in V'$ ($v'' \in V''$), для которых не существует вершины $v'' \in V''$ ($v' \in V'$) такой, что $v'_{tr} = v''_{tr}$, также добавляются во множество вершин результирующего графа. Аналогично выполняется объединение рёбер.

Последний шаг в построении модели: объединить графы (V_l, E_l), полученные $\forall l \in L$, в один. Для этого каждой вершине $v \in V_l$ (ребру $(u, v) \in E_l$) ставится дополнительно в соответствие метка языка v_{lang} $((u, v)_{lang})$. Готовая модель представляет собой граф $(\mathcal{V}, \mathcal{E})$, который получается следующим образом:

$$\mathcal{V} = \bigcup_{l \in L} V_l$$

$$\mathcal{E} = \bigcup_{l \in L} E_l$$

Классификация происходит декомпозицией сообщения на триграммы, вычислением функции схожести сообщения с языком и выбором языка с максимальным значением этой функции. Формально, если $t_1, t_2, \ldots, t_n$ - триграммы сообщения new_msg в порядке их вхождения, то результатом будет

$$\operatorname*{argmax}_l score_l,$$

где вектор $score$ длины $|L|$ определяется как:

$$score_l = \sum_{i=1}^{n} get\_v(t_i, l) + \sum_{i=1}^{N-1} get\_e(t_i, t_{i+1}, l)$$

$$get\_v(tr, l) = \begin{cases} \dfrac{v_{cnt}}{\sum_{w \in \mathcal{V}} w_{cnt}}, & if\ \exists v \in \mathcal{V} : v_{tr} = tr \\ & \quad v_{lang} = l \\ 0, & otherwise \end{cases}$$

$$get\_e(tr_1, tr_2, l) = \begin{cases} \dfrac{(u, v)_{cnt}}{\sum_{(w,q) \in \mathcal{E}} (w, q)_{cnt}}, & if\ \exists (u, v) \in \mathcal{E} : u_{tr} = tr_1 \\ & \quad v_{tr} = tr_2 \\ & \quad (u, v)_{lang} = l \\ 0, & otherwise \end{cases}$$

Такая модель позволяет зафиксировать одновременно частотность триграмм и их положение, при этом все триграммы из документов участвуют в классификации. Например, неправильное написание какого-то слова, которое может быть не учтено n-граммным подходом (поскольку он учитывает лишь наиболее часто встречающиеся сочетания), добавит определённости при определении языка методом LIGA.

Плюсы и минусы:

+ Высокая скорость обучения и классификации

+ Максимально использует информацию из обучающей выборки

+ Возможность дообучения «на лету»

+ Возможность визуализации модели

− Большой объём модели и, как следствие, относительно высокие требования к памяти

3 Исследование и построение решения задачи

3.1 Сравниваемые системы

3.1.1 TextCat

TextCat[5] — реализация подхода Кавнара и Тренкля [1], основанного на n-граммах. Поскольку этот метод был одним из первых, практически все появляющиеся системы сравниваются именно с ним. Оригинальные модели языков достаточно устарели, но, к счастью, у программы есть возможность обучения.

3.1.2 Google CLD

Google compact language detector[6] — программный продукт, встроенный в браузер Chromium, предназначенный для определения языка просматриваемой страницы. Использует 4-граммы и умеет считывать метаинформацию о веб-страницах. Впрочем, для коротких текстов он тоже используется, например, в сервисе Google Translate. Система поставляется обученной на огромном корпусе текстов на более чем 100 языках.

3.1.3 Langid.py

Программа langid.py[7] за авторством Луи и Болдуина [7], выпущенная в 2011 году, позиционируется как быстрое и точное решение задачи определения языка текста. Продукт поддерживает 97 языков.

3.1.4 LIGA

Реализация метода LIGA, выполненная в рамках курсовой работы в соответствии со статьёй [2]. Подход подробно описан в секции 3.4.

[5] http://odur.let.rug.nl/vannoord/TextCat/

[6] https://code.google.com/p/cld2/

[7] https://github.com/saffsd/langid.py

3.1.5 Предлагаемое улучшение LIGA

Я предлагаю улучшенную версию алгоритма LIGA — LIGAv2. Основное отличие заключается в изменении функций get_v и get_e:

$$get\_v\_v2(tr, l) = \begin{cases} \dfrac{(\log \frac{|L|}{|\{r \in L | \exists v \in \mathcal{V} : v_{tr} = tr, v_{lang} = r\}|} + 1) \times v_{cnt}}{\sum_{w \in \mathcal{V},\ w_{lang} = l} w_{cnt}}, & if\ \exists v \in \mathcal{V} : v_{tr} = tr \\ & \quad v_{lang} = l \\ 0, & otherwise \end{cases}$$

$$get\_e\_v2(tr_1, tr_2, l) = \begin{cases} \dfrac{(\log \frac{|L|}{edges} + 1) \times (u, v)_{cnt}}{\sum_{(w,q) \in \mathcal{E},\ (w,q)_{lang} = l} (w, q)_{cnt}}, & if\ \exists (u, v) \in \mathcal{E} : u_{tr} = tr_1 \\ & \quad v_{tr} = tr_2 \\ & \quad (u, v)_{lang} = l \\ 0, & otherwise \end{cases}$$

$$edges = |\{r \in L \mid \exists (u, v) \in \mathcal{E} : u_{tr} = tr_1, v_{tr} = tr_2, (u, v)_{lang} = r\}|$$

Мотивация к такому изменению следующая — гораздо полезнее знать, насколько специфична данная триграмма данному языку, чем то, сколько раз она встречалась в обучающей выборке вообще. Для этого в знаменателе общая сумма очков всех вершин графа (аналогично для рёбр) заменена на сумму очков всех вершин подграфа (V_l, E_l). Также вводится логарифмический коэффициент, который характеризует «уникальность» данной триграммы в обучающей выборке. Логарифм берётся от общего количества языков $|L|$, делённого на количество языков, в которых данная триграмма встретилась. Таким образом получается, что сочетания, встретившиеся, например, лишь в одном языке, дают несколько больший вклад этому языку, чем сочетания, присутствовавшие во всех языках обучающей выборки.

После таких изменений появляется шанс, что фраза *"Привіт, груша"* будет корректно распознана как украинская, поскольку триграммы *віт* и *иві* будут иметь больший вес, чем остальные.

3.1.6 LogR

Реализация метода LogR, выполненная в рамках курсовой работы в соответствии со статьёй [3]. Общий подход описан в секции 3.3. Данная версия использует логистическую регрессию из пакета LibLinear [6], а также следующие признаки:

- Логарифм количества вхождений каждой уни-, би-, три- и квадграммы.
- Имя оставившего твит пользователя Twitter, его местоположение и отображаемое имя, а также их префиксы

- Индикатор: написано ли имя пользователя латиницей
- Встреченные в твите хэштеги [8]
- Встреченные в твите упоминания [9]
- Доменное имя 1 и 2 уровней, а также протокол для всех ссылок, встреченных в сообщении (при этом, если ссылка была сокращённой, например bit.ly/something, то происходит http-запрос для определения конечной цели)

3.2 Обучающий корпус

Твиты для обучающего корпуса собирались из следующих мест:

1. **Из статей по теме распознавания языка в Twitter.** В работе [2] тексты сообщений оказались в открытом доступе, около 9 тысяч на западноевропейских языках. Также авторы статьи [3] предоставили собранный корпус, в котором около 6 тысяч твитов, содержащих помимо текста ещё и метаинформацию о пользователях. Огромный набор текстов на русском языке прилагается к статье [5] - более двухсот тысяч твитов.

2. **Извлечённые с помощью TwitterAPI по идентификационному номеру.** Согласно политике Twitter, в открытом доступе могут находиться не тексты сообщений, а лишь их идентификационные номера. Вместе с работой [4] распространяется архив, содержащий номера твитов, используемых в ней при тестировании.

3. **С сайта Indigenous Tweets.**[10] Данная веб-страница содержит список малопредставленных в Twitter языков, а также перечисление всех известных пользователей с указанием количества сообщений, которое они оставили на конкретном языке. Были выделены пользователи, пишущие в основном на татарском и чувашском языках, а их сообщения были выкачаны с помощью TwitterAPI. Таким образом было получено 6 тысяч твитов.

[8] Хэштег - это последовательность символов без пробелов, начинающаяся с '#', например: *#HarryPotter*, *#russia2014*

[9] Упоминание - это отображаемое имя пользователя, предварённое символом '@', например: *@KremlinRussia*

[10] http://indigenoustweets.com/

4. **С помощью собственного аккаунта.** Имея аккаунт в Twitter, можно подписаться на новостные рассылки и скачивать их. Таким образом было извлечено 800 твитов на турецком. Такое малое количество обусловлено ограничениями TwitterAPI.

Итоговое распределение по языкам показано в таблице 1 в приложении 1.

3.3 Описание сценариев нормализации

Сообщения в социальных сетях часто содержат ошибки, выражения эмоций, неправильное употребление слов и знаков препинания. Чтобы классификатор был устойчивым к такого рода явлением, нужно *нормализовывать* каждое сообщение — то есть приводить его к какому-то стандартному виду. Для того, чтобы проследить зависимость качества классификации от степени нормализации текстов, были использованы следующие сценария нормализации:

1. Удаляются отметки о ретвитах[11], ссылки, упоминания, хэштеги, символы пунктуации заменены на пробел, цифры удалены, последовательно идущие пробельные символы заменены на один пробел, текст переведён в нижний регистр

2. Всё то же, что и в пункте один, плюс: все последовательности из трёх и более подряд идущих одинаковых символов заменены на два символа (например, слово *ооооооо* превращается таким образом в *оо*, что помогает сгладить последствия проявления эмоций), все слова из 2 и менее букв удалены

В обоих случаях сообщения, в которых после нормализации не осталось никаких символов, не включались в итоговую выборку.

3.4 Выбор меры качества классификации

Классификаторы сравнивались по общепринятым (см. [8]) мерам качества мультиклассовой классификации: точности, полноте, F1-мере, accuracy.

3.5 Результаты тестирования

Тестирование проводилось при помощи одной из разновидностей метода *кросс-валидации*. Обучающая выборка каждого языка делилась на

[11] Ретвит — твит, отправитель которого не является автором, при этом обычно в тексте идёт имя автора после букв RT, например, RT @KremlinRussia

три части: по M случайных сообщений в тестовой и тренировочной части, все оставшиеся — в третьей. Те классификаторы, у которых есть возможность обучиться на имеющихся данных, обучались на тренировочной части. Затем собирались результаты предсказаний языка на тестовой части выборки, вычислялись точность, полнота, F1-мера и accuracy. Показатели вычислялись в соответствии с работой [8]. Такое разбиение происходило 10 раз для каждого классификатора, а полученные результаты усреднялись.

3.5.1 Зависимость качества классификации от количества обучающих сообщений

Из таблиц 2, 3 и 4 приложения 2 видно, что F1-мера всех классификаторов изменяется не более чем на 1%.

3.5.2 Зависимость качества классификации от степени нормализации

Из сравнения таблиц 4 и 5 приложения 2 видно, что изменение качества работы лишь двух классификаторов — *langid* и *TextCat* — составило более одного процента в худшую сторону. Отсюда следует, что второй сценарий нормализации не имеет преимуществ над первым

3.5.3 Примеры ошибок

Выявлены следующие причины возникновения ошибок (в скобках после сообщений указан сначала предполагаемый язык, а затем предсказанный):

1. Похожесть языков: *lang leve ikea* (nl → en), *state department condemns concerted campaign intimidate international journalists cairo* (en → fr). Некоторые языки с родственным алфавитом похожи. Не имеющий знаний об арабских языках человеку сложно отличить один арабский язык от другого. Хоть применение машинного обучения и помогает в этом, но некоторые конструкции всё же не поддаются различию.

2. Наличие имён собственных на другом языке: *moyes ponders toffees selection everton boss david moyes will have decisions make both ends the pitch* (en → fr). Названия групп, фирм, мест — всё это накладывает свой отпечаток на сообщение, и в этом случае классификаторам легко ошибиться даже на достаточно длинных сообщениях.

3. Фразы на другом языке: *харесах видеоклип watch mario balotelli failed backheel* (bg → en). В таких твитах сложно определить, какой язык

является первичным. Например, некоторые твиты на языке урду представляют из себя записи вида примерно «текст на английском: перевод этого же текста на урду». Однозначный вывод сделать автоматически очень сложно. Возможно, в этом помог бы анализ метаинформации твита.

4. Неверная разметка: *встиг підстригтися уже наступний раз записався* (ru → uk). Некоторые тексты, особенно среди двухсоттысячного набора русских твитов, на самом деле не написаны на предполагаемом языке. К счастью, их не очень много и они мало влияют на оценку качества, тем более что все классификаторы на них «ошибаются»

5. Малая длина: *кис*, *прекратите*, *lesles*, *haa dus*, *lool grave*. Тексты этой категории сложно отнести к конкетному языку даже живому человеку. При ужесточении нормализации таких коротких твитов становится больше

3.5.4 Выводы

Все классификаторы, кроме LogR, показали конкурентоспособные результаты. Причиной такого неуспеха LogR могло стать отсутствие настройки параметров основного шага метода — логистической регрессии, она запускалась со стандартными параметрами.

Качество классификации несущественно изменяется с увеличением в два и три раза тренировочной выборки.

Первый сценарий нормализации является более оптимальным для работы классификаторов, чем второй. Это значит, что удаляемые вторым вариантом части сообщения не мешают определению языка, а, наоборот, помогают.

Предложенный подход LIGAv2 в среднем на 19.5% обгоняет по F1-мере другие системы. По этой же характеристике он на 4% впереди оригинального LIGA.

4 Описание практической части

4.1 Язык программирования

Все программы выполнены на языке *Python 2*. Это интерпретируемый язык, то есть он может быть запущен на любом компьютере, где установлен интерпретатор Python. Также для него есть множество библиотек. Так, например, две из трёх существующих сравниваемых систем — Google CLD2 и TextCat — распространяются для Python в качестве таких библиотек, а третья система — langid — сама написана на этом языке. Также TwitterAPI доступен в качестве библиотеки для Python.

4.2 Архитектура решения

4.2.1 /scripts

Здесь находятся программы, обрабатывающие твиты и проводящие тестирование. Из них стоит упомянуть следующие:

- gen_all_texts.sh — процедура извлечения из твитов текста. Записывает в папку /parsed_text сначала файлы, полученные при извлечении текста из различных источников, а затем объединяет файлы, соответствующие одному языку, в один, где каждый твит записан ровно на одной строке. Здесь же происходит нормализация сообщений, а также сбор статистики количества твитов на каждом языке — файл STATISTICS в полученной директории.

- gen_all_texts_features.sh — аналогичен предыдущему, но твиты складываются в папку /parsed_text_features, и в них присутствует информация, разделённая на колонки символом табуляции в следующем формате (по порядку слева направо):

  1. Нормализованный текст
  2. Имя пользователя
  3. Отображаемое имя пользователя
  4. Местоположение
  5. Список хэштегов (через запятую), присутствовавших в сообщении
  6. Список упоминаний (через запятую), присутствовавших в сообщении
  7. Список ссылок (через запятую), присутствовавших в сообщении

При отсутствии информации из пунктов 2-4 в файл пишется 'not-given', при отсутствии список 5-7 — пустая строка.

- string_processing.py — в этом файле находятся все функции, отвечающие за нормализацию текста. Изменяя его и перезапуская gen_all_texts, можно экспериментировать со степенью обработки сообщений.

- twitter_processing.py — функции, с помощью которых проводилось извлечение данных из Twitter.

- scripts.py — различные вспомогательные подпрограммы для тестирования

- main.py — программа, использовавшаяся для тестирования имеющихся решений. Предполагает наличия нескольких установленных библиотек. Для каждого классификатора производит кросс-валидацию, записывает в один файл результаты, в другой — все неправильно классифицированные сообщения.

4.2.2 /programs

Здесь находятся реализованные в рамках курсовой работы методы решения задачи определения языка в Twitter в виде классов — LIGAv2 (класс LIGA в файле liga/liga.py), LIGA (класс LIGA в файле liga/liga_original.py), LogR (класс LogR в файле logr/logr.py). Все три подхода предполагают при инициализации передачу в качестве параметра папки, в которой находятся файлы с тренировочной выборкой. Метки языка берутся из названий файлов.

Также во всех трёх присутствует метод classify(tweet), который выдаёт результат классификации для переданного твита. Для LIGA и LIGAv2 твит должен быть записан в виде текста, а для LogR — в формате, описанном в разделе 4.2.1.

Кроме того, у обеих реализаций LIGA есть метод classify_file(sep, path), который выдаёт список предсказаний для твитов, записанных в файле по пути path_to_file и разделённых между собой строкой separator.

5 Заключение

В рамках данной работы предполагалось изучить актуальные методы решения задачи автоматического определения языка сообщений социальной сети Twitter, провести сравнительное тестирование некоторых, исследовать возможность улучшения качества классификации.

Поставленные цели были достигнуты и были получены следующие результаты:

1. Реализовано 2 подхода к решению этой задачи на языке программирования *Python*, а также модифицированная версия одного из них
2. Собран корпус твитов из более чем двухсот тысяч сообщений
3. Написан комплекс скриптов для проведения совместного тестирования полученных решений с программными продуктами, успешно используемыми в области обработки естественых языков
4. Проведено тестирование, сделаны выводы на основе результатов

Была исследована зависимость качества классификации от степени нормализации сообщений, рассмотрены примеры ошибочно определённых твитов, сделаны предположения по поводу причин ухудшения качества работы классификаторов.

Тестирование показало, что улучшенная версия метода LIGA превосходит по качеству классификации все остальные методы.

Список литературы

[1] Cavnar W. B. et al. N-gram-based text categorization //Ann Arbor MI. – 1994. – Т. 48113. – №. 2. – С. 161-175.

[2] Tromp E., Pechenizkiy M. Graph-based n-gram language identification on short texts //Proc. 20th Machine Learning conference of Belgium and The Netherlands. – 2011. – С. 27-34.

[3] Bergsma S. et al. Language identification for creating language-specific Twitter collections //Proceedings of the Second Workshop on Language in Social Media. – Association for Computational Linguistics, 2012. – С. 65-74.

[4] Carter S., Weerkamp W., Tsagkias M. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text //Language Resources and Evaluation. – 2013. – Т. 47. – №. 1. – С. 195-215.

[5] Ю.В. Рубцова. Метод построения и анализа корпуса коротких текстов для задачи классификации отзывов // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XV Всероссийской научной конференции RCDL'2013, Ярославль, Россия, 14-17 октября 2013 г. – Ярославль: ЯрГУ, 2013. –С. 269-275.

[6] Fan R. E. et al. LIBLINEAR: A library for large linear classification //The Journal of Machine Learning Research. – 2008. – Т. 9. – С. 1871-1874.

[7] Baldwin T., Lui M. Language identification: The long and the short of the matter //Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. – Association for Computational Linguistics, 2010. – С. 229-237.

[8] Sokolova M., Lapalme G. A systematic analysis of performance measures for classification tasks //Information Processing & Management. – 2009. – Т. 45. – №. 4. – С. 427-437.

Приложение 1 Распределение твитов по языкам

Язык	Количество твитов
Арабский	1171
Английский	2234
Болгарский	1880
Испанский	2350
Итальянский	1538
Маратхи	1154
Немецкий	2208
Непальский	1678
Нидерландский	2032
Персидский (фарси)	2361
Русский	196836
Татарский	3248
Турецкий	781
Украинский	627
Урду	1073
Французский	2239
Хинди	1209
Чувашский	2584
Всего	227203

Таблица 1: Распределение по языкам твитов обучающей выборки.

Приложение 2 Результаты экспериментов

Мера	Google CLD2	langid.py	LIGAv2	LIGA	TextCat	LogR
Точность	85.2%	78.5%	93.5%	90.3%	93.4%	29.4%
Полнота	79.3%	78.8%	93.2%	89.0%	90.0%	31.3%
F1-мера	82.2%	78.7%	93.3%	89.7%	91.7%	30.3%
Accuracy	79.4%	78.7%	93.4%	89.2%	90.0%	29.5%

Таблица 2: Показатели качества классификации при $M = 250$.

Мера	Google CLD2	langid.py	LIGAv2	LIGA	TextCat	LogR
Точность	85.2%	78.4%	94.1%	91.2%	93.6%	27.2%
Полнота	79.5%	78.7%	93.9%	90.2%	90.3%	29.1%
F1-мера	82.3%	78.6%	94.0%	90.7%	91.9%	28.1%
Accuracy	79.3%	78.8%	93.6%	89.9%	90.3%	28.9%

Таблица 3: Показатели качества классификации при $M = 500$.

Мера	Google CLD2	langid.py	LIGAv2	LIGA	TextCat	LogR
Точность	85.2%	78.4%	94.3%	90.8%	93.8%	29.5%
Полнота	79.4%	78.7%	94.0%	89.3%	90.5%	29.9%
F1-мера	82.2%	78.6%	94.2%	90.1%	92.1%	29.7%
Accuracy	79.5%	78.5%	94.1%	89.6%	90.5%	29.4%

Таблица 4: Показатели качества классификации при $M = 700$.

Мера	Google CLD2	langid.py	LIGAv2	LIGA	TextCat	LogR
Точность	85.2%	76.7%	93.6%	90.2%	92.2%	28.0%
Полнота	78.3%	76.3%	93.4%	88.7%	87.6%	30.0%
F1-мера	81.6%	76.5%	93.5%	89.5%	89.8%	28.9%
Accuracy	78.5%	76.3%	93.4%	88.8%	87.4%	30.6%

Таблица 5: Показатели качества классификации при $M = 700$ и втором сценарии нормализации.