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

1 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 база из сотен документов на нескольких языках, то можно «предсказать» язык поступившего на рассмотрение документа, сравнив его с имеющимися. В общем случае, нужно по имеющимся данным построить так называемую модель, а затем все действия с текстами проводить в терминах этой модели. Классическим примером является метод, описанный в 1994 году: каждому документу сопоставим «профиль документа» — упорядоченный по числу встречаний 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. Составляется профиль этого документа
2. Сравнивается с имеющимися профилями языков
3. Тот язык, чей профиль наиболее похож на профиль документа, объявляется результатом

Вышеописанный метод плохо работает для коротких сообщений. В то же время, поток информации в виде коротких шумных сообщений нельзя игнорировать — в социальной сети Twitter среднее количество сообщений в день составляет примерно 58 000 000, 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 полноте распознавания языка полученного результата над существующими.

2 Постановка задачи

2.1 Формальное описание задачи автоматического определения языка

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

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

(здесь $msg_i, i \in \overline{1..N}$, - 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.

2.2 Цели и задачи курсовой работы

В данной работе в качестве текстов выступают так называемые «твиты» - сообщения из социальной сети Twitter[1], а множество L соответствует 18 языкам, которые можно разделить на несколько групп по типу алфавита:

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

Цели работы:

1. Исследовать современные решения задачи автоматического определения языка коротких сообщений
2. Провести совместное сравнительное тестирование некоторых методов решения задачи и выяснить, действительно ли они показывают заявленное авторами качество классификации

[1] https://twitter.com/

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 800 твитов для каждого языка

2. Реализовать один или несколько методов решения задачи автоматического определения языка коротких сообщений

3. Организовать удобный интерфейс для автоматического тестирования методов

4. Выбрать меру качества классификации

3 Обзор существующих решений

3.1 Подход, основанный на частоте 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 $|i - j|$.

Cavnar & Trenkle показали, что при значении $K \approx 300$, составленные профили языков очень хорошо эти языки характеризуют. Также они предлагают использовать длину n-грамм вплоть до 5.

Этим подходом пользуются многие программные продукты, например, *langid*, *TextCat*, *Google CLD*, о которых речь пойдёт в части 4.

Плюсы и минусы:

+ Малый размер модели
+ Высокая скорость обучения и классификации
− Качество классификации сильно зависит от длины текста
− Много информации из документов не попадает в модель

3.2 Prediction by partial matching

Кодируются все имеющиеся документы, основываясь на частотах встречания цепочек из не более чем n символов (чем больше частота — тем короче код входящих в цепочку символов). Затем для new_msg вычисляется длина кода при помощи полученного кодирования и выбирается наиболее компактно кодирующий язык.

Такой подход предлагается, например, в [3]

Плюсы и минусы:

+ Мало параметров

+ Высокая точность классификации

− Для нормального функционирования нужны корпуса текстов, измеряемые мегабайтами

3.3 Подходы, использующие «стандартные» алгоритмы машинного обучения

Из документов вычленяются признаки, которые их характеризуют, и каждому документу сопоставляется вектор признаков. Далее применяется один из алгоритмов машинного обучения (например, логистическая регрессия или метод опорных векторов) для построения модели зависимости целевой переменной (языка сообщения) от вектора признаков. New_msg при классификации точно также заменяется на вектор признаков, а затем классификация проходит согласно выбранному алгоритму.

Благодаря гибкости в выборе признаков, подход является перспективным по отношению к определению языка твитов. Так, например, в работах [3] [4] используются, помимо текстовых, такие признаки: имя пользователя в Twitter; 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:

+ Хорошо разработанный набор стандартных алгоритмов

+ Возможность настройки для выбранной предметной области

− Нужно тщательно подбирать признаки и алгоритм, дабы получить адекватный результат

3.4 Language identification graph-based approach (LIGA)

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 ($v_{tr} \neq u_{tr}\ \forall u \neq v$), 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 раз, когда такое происходило.

Далее, объединим полученные графы 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

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

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

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 разбиением на триграммы, а затем «укладыванием» этих триграмм на пути графа. 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:

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

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

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

Метод предложен в работе [2]. 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, будучи упущенным в n-граммном подходе (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

4 Подготовка к тестированию

4.1 Сравниваемые системы

4.1.1 TextCat

TextCat[2] — авторская реализация подхода Canvar & Trenkle [1], основанного на n-граммах. Этот классический продукт часто берётся в качестве стандарта при сравнении алгоритмов определения языка. Оригинальные модели языков достаточно устарели, но, к счастью, у программы есть возможность обучения.

4.1.2 Google CLD

Google compact language detector[3] — 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,

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

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

например, в сервисе Google Translate. Система поставляется обученной на огромном корпусе текстов на более чем 100 языках.

4.1.3 Langid.py

Программа langid.py[4] за авторством Lui & Baldwin [7], выпущенная в 2011 году, позиционируется как быстрое и точное решение задачи определения языка текста. Продукт поддерживает 97 языков.

4.1.4 LIGA

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

4.1.5 LIGAv2

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

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

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

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

Мотивация к такому изменению следующая. Гораздо полезнее знать, насколько специфична данная триграмма данному языку, чем то, какой процент 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, идея которого навеяна idf[5], и который характеризует «уникальность» данной триграммы в обучающей выборке.

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.

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

[5] inverse document frequency

4.1.6 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 частоты встречания 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 [6]
- Встреченные в твите упоминания [7]
- 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)

4.2 Обучающий корпус

Твиты для обучающего корпуса собирались из следующих мест:

1. **Приложенные к статьям.** В работе [2] тексты сообщений оказались в открытом доступе, около 9 тысяч штук на европейских языках. Также авторы статьи [3] поделились собранным корпусом, 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.

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

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

3. **С сайта Indigenous Tweets.**[8] 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.

4. **С помощью собственного аккаунта.** В распоряжении оказался аккаунт турецкого пользователя, подписанного на новостные рассылки на турецком языке. Было извлечено 800 твитов на турецом. Такое малое количество обусловлено ограничениями TwitterAPI.

Итоговое распределение по языкам (после нормализации по схеме 2) показано в таблице 1.

4.3 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. Дабы проследить зависимость качества классификации от степени нормализации текстов, были использованы следующие сценария нормализации:

1. Из твита удаляются отметки о ретвитах[9], ссылки, упоминания, хэштеги, символы пунктуации заменены на пробел, цифры удалены, последовательно идущие пробельные символы заменены на один пробел

2. В дополнение к пункту 1: все последовательности из трёх и более подряд идущих одинаковых символов заменены на два символа (например, слово *ооооооoy* превращается таким образом в *ooy*), все слова из 2 и менее букв удалены, текст приведён в нижний регистр.

В обоих случаях после нормализации сообщения, в которых не осталось никаких символов, выбрасывались из обучающей выборки.

[8] http://indigenoustweets.com/

[9] Например, RT @KremlinRussia

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.

4.4 Выбор меры качества классификации

В данной работе в качестве меры качества используется *полнота*, то есть доля правильно определённых сообщений на фиксированном языке от общего количества твитов на этом языке.

Выбор обусловлен тем, что классификатор LIGA, ввиду большого числа обучающих данных на русском языке, выдаёт почти всем кириллическим текстам метку «русский». В то же время, на несколько десятков, например, болгарских сообщений он выдаёт «болгарский». Если посчитать *точность* определения болгарского языка — отношение числа правильно определённых болгарских твитов к числу твитов, определённых как болгарские — то она будет близка к 100%. Ожидаемо, точность распознавание русского языка при этом падает. Но, так как русских сообщений на два порядка больше, то падает она примерно на 1%. Таким образом, высокие показатели *точности* на деле скрывают неудовлетворительные результаты классификации.

В то же время, полнота соответствует интуитивному понятию «точности».

5 Результаты тестирования

Тестирование проводилось методом *кросс-валидации* в 10 итераций.

К сожалению, реализация LogR обучалась на два порядка дольше других классификаторов (во многом из-за того, что делала много http-запросов на раскрытие ссылок), а при уменьшении числа твитов в обучающей выборке показывала неудовлетворительный результат, близкий к случайной выдаче языка, поэтому не вошла в итоговые таблицы и графики.

Прочерки в таблицах означают, что классификатор ни разу не выдал в качестве ответа такой язык.

5.1 Зависимость полноты классификации от количества данных, даваемых на обучение

Таблица 2 показывает, что вышеуказанной зависимости не наблюдается.

5.2 Зависимость полноты классификации от степени нормализации

На таблице 3 видно, что в полнота, в целом, падает при использовании сценария 2 для нормализации. Впрочем, стоит заметить, что алгоритм

Язык	Google CLD2	langid.py	LIGAv2	LIGA	TextCat
Болг.	81/81/81/82	77/77/77/77	94/94/94/94	0/0/0/0	88/88/88/87
Рус.	91/91/91/91	86/86/86/86	90/90/90/90	99/99/99/99	87/87/87/87
Татар.	-/-/-/-	-/-/-/-	99/99/99/99	00/00/00/00	99/99/99/99
Укр.	80/82/81/81	88/88/88/88	76/78/77/79	-/-/-/-	88/88/88/89
Чуваш.	-/-/-/-	-/-/-/-	98/99/98/99	00/00/00/00	96/96/96/96
Араб.	84/82/84/84	90/90/90/90	97/97/98/98	89/89/90/89	95/95/94/95
Перс.	87/87/87/88	89/89/89/88	94/94/94/94	95/95/95/95	88/88/88/88
Урду	89/89/89/89	90/90/89/90	97/96/96/96	81/81/82/82	92/92/92/92
Англ.	97/97/97/97	96/96/96/96	87/88/88/87	92/92/92/92	93/93/93/93
Испан.	86/86/86/85	93/93/93/93	97/96/97/96	97/96/96/96	91/91/91/91
Итал.	92/92/92/92	98/98/98/98	98/98/98/98	92/92/92/92	96/96/97/96
Нем.	94/94/94/94	96/96/96/97	97/97/97/97	95/95/95/95	94/93/93/93
Нидер.	90/90/91/90	96/96/96/96	98/98/98/98	96/96/96/96	94/94/94/94
Турец.	88/88/89/88	89/90/89/89	94/94/94/93	14/14/14/15	89/89/89/88
Франц.	91/92/91/92	96/96/96/96	96/96/96/96	97/98/97/98	93/93/93/93
Марат.	91/91/91/91	80/81/80/82	92/91/91/91	73/73/73/74	81/82/80/80
Непал.	88/88/88/88	71/71/70/70	92/93/92/92	93/93/94/93	81/82/83/83
Хинди	97/97/97/97	83/83/84/85	95/95/95/95	85/86/86/86	82/83/81/82

Таблица 2: Полнота классификации при обучении на 50/60/70/80% от общего числа твитов на данном языке.

LIGAv2 преодолел барьер в 80% для украинского языка.

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).

Язык	Google CLD2	langid.py	LIGAv2	LIGA	TextCat
Болгарский	81/84	77/70	94/92	00/00	88/87
Русский	91/93	86/86	90/94	99/99	87/86
Татарский	-/-	-/-	99/99	00/00	99/99
Украинский	81/83	88/87	77/84	-/-	88/83
Чувашский	-/-	-/-	98/90	00/00	96/95
Арабский	84/78	90/92	98/98	90/90	94/93
Персидский	87/80	89/87	94/92	95/95	88/84
Урду	89/85	89/79	96/93	82/63	92/88
Английский	97/95	96/92	88/85	92/88	93/90
Испанский	86/81	93/84	97/94	96/93	91/82
Итальянский	92/90	98/95	98/98	92/84	97/95
Немецкий	94/93	96/96	97/97	95/97	93/91
Нидерландский	91/90	96/93	98/97	96/91	94/90
Турецкий	89/91	89/88	94/94	14/27	89/86
Французский	91/88	96/92	96/95	97/95	93/97
Маратхи	91/90	80/83	91/94	73/79	80/75
Непальский	88/88	70/70	92/93	94/94	83/80
Хинди	97/93	84/72	95/85	86/64	81/75

Таблица 3: Полнота классификации при использовании первого/второго сценария нормализации.

Некоторые языки с родственным алфавитом похожи. Непосвящённый человек не отличит 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 «learn new word — self — in urdu: ...». Однозначный вывод сделать автоматически очень сложно. Возможно, в этом помог бы анализ метаинформации о пользователе.

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

5.4 Выводы

Оригинальный алгоритм LIGA вырождается в константный классификатор на кириллических языках (поскольку выборка русского языка очень велика) и очень плохо определяет турецкий язык (его выборка, напротив, самая маленькая). Предлагаемое улучшение — LIGAv2 — не показывает таких проблем.

Из таблицы 3 видно, что арабские и латинские языки LIGAv2 классифицирует намного лучше других продуктов, в то время как на остальных языках он не сильно отстаёт. Это доказывает состоятельность метода и успешность применённого подхода.

6 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 модернизированная версия одного из них

2. Собрана корпус 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 по поводу 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 не уступает, а зачастую и превосходит по полноте 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.