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 три группы по типу алфавита:

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

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
3. Исследовать зависимость качества классификации от различных факторов: степени нормализации сообщений, мощности обучающей выборки, возможных предположений о структуре текстов

[1] https://twitter.com/

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 метрику 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 ко-

роче 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 TODO

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 TODO 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:

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

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

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

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

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

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

4.1.3 Langid.py

Программа langid.py[4] за авторством Lui & Baldwin, 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 мной в соответствии с работой TODO. Подход подробно описан в секции 3.4.

4.1.5 LogR

Реализация метода LogR из работы TODO. Общий подход описан в секции 3.3. Данная версия использует 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 [5]
- Встреченные в твите упоминания [6]
- 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] https://github.com/saffsd/langid.py

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

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

4.1.6 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[7], 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.

4.2 Обучающая выборка

5 Примеры работы программы

5.1 Пример 1

В данной системе матрицы A, B, P, X_0 являются единичными матрицам в $\mathbb{R}^3$, векторы p, x_0 — нулевыми. Диапазон времени: $t_0 = 0$, $t_1 = 3$, фазовые ограничения отсутствуют. За статичные направления l_1 и l_2 взяты векторы $[1, 0, 0]$ и $[0, 1, 0]$, за динамичные — $l_1(t) = [\sin(t); \cos(t); t], l_2(t) = [\cos(t); \sin(t); t]$. %endcenter

[7] inverse document frequency

5.2 Пример 2

В данной системе:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1, \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1, \end{pmatrix} P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1, \end{pmatrix}$$

$$x_0 = [0, 0, 0]^T, p = [0, 0, 0]^T, t_0 = 0, t_1 = 3.$$

В данной системе есть фазовые ограничения $x_1 \leqslant 3$.

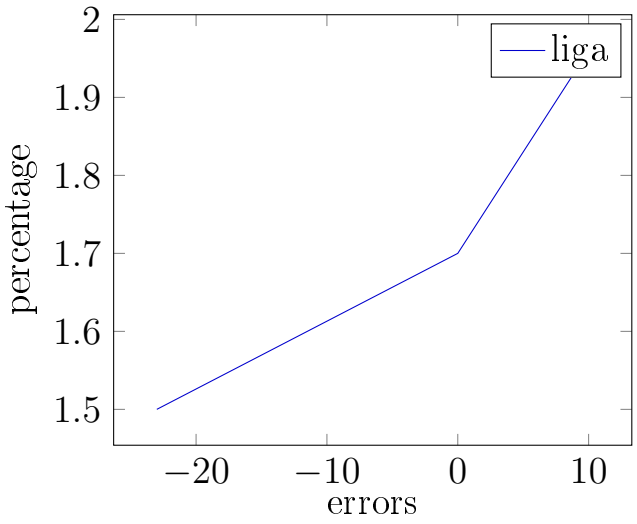
5.3 Пример 3

Рассмотрим колебательную систему из задания прошлого семестра. В этой системе:

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2\frac{g}{l_1} & \frac{g}{l_1} & 0 & 0 \\ 2\frac{g}{l_2} & -2\frac{g}{l_2} & 0 & 0, \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1, \end{pmatrix} P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1, \end{pmatrix}$$

$$x_0 = [0.1, 0.3, 1, 1]^T, p = [0, 0, 0, 0]^T, t_0 = 0, t_1 = 3, l_1 = 2, l_2 = 1.$$

На систему наложены фазовые ограничения $|x_1| \leqslant y_1$, $y_1 = 1$.



Проекция множества достижимости на статическую плоскость (l_1, l_2).

Проекция трубки достижимости на статическую плоскость (l_1, l_2).

Проекция трубки достижимости на статическую плоскость (l_1, l_2).

Из рисунков трубки достижимости видно, что фазовые ограничения выполняются.

6 Библиография

Список литературы

[1] Голубев Ю. Ф. Основы теоретической механики: Учебник. 2-е изд., перераб. и дополн. — М.:Изд-во МГУ, 2000.

[2] P. Gagarinov, Alex A. Kurzhanskiy Ellipsoial toolbox: ver. 2.0 beta 1, 2013.