GoTo Data Science Challenge 2016

Григорьев Георгий

19 октября 2016 г.

/Наука и образование
  /Наука и образование/наука
    /Наука и образование/наука/математика
    /Наука и образование/наука/физика
    /Наука и образование/наука/химия
    /Наука и образование/наука/информатика
      /Наука и образование/наука/информатика/биоинформатика
      /Наука и образование/наука/информатика/анализ данных
    /Наука и образование/наука/литература
    /Наука и образование/образование
      /Наука и образование/образование/школьное
      /Наука и образование/образование/высшее
      /Наука и образование/образование/дополнительное
        /Наука и образование/образование/дополнительное/GoTo
/Политика
  /Политика/Внутренняя
  /Политика/Внешняя
/Экономика и бизнес
  /Экономика и бизнес/Бизнес
    /Экономика и бизнес/Бизнес/Стартапы
      /Экономика и бизнес/Бизнес/Стартапы/E-Contenta
    /Экономика и бизнес/Бизнес/Крупные компании
  /Экономика и бизнес/Экономика
/Отдых и развлечения
  /Отдых и развлечения/Кино
  /Отдых и развлечения/Театр
  /Отдых и развлечения/Компьютерные игры
/Здоровье и красота/Фитнес
/Здоровье и красота/Медицина
/Здоровье и красота/Косметология

Введение

Задача оказалась сложнее, чем кажется на первый взгляд. В ходе выполнения работы я улучшил свои познания в некоторых областях компьютерных наук. Это был интересный опыт, который стоит развивать дальше. *(Ну конечно, ведь я наметил себе будущую профессию)*

К сожалению, численная оценка методов представлена лишь в 3 методе, потому как лишь тогда я смог найти размеченный датасет для обучения(и, соответственно, для проверки). Не стал вдаваться во всякие AP@K и AUG Score.

Цель и средства

Цель:

Попрактиковаться в создании моделей машинного обучения.

Средства:

Python 3.5, Jupyter Notebook 4, macOS Sierra, Pages 6.0,
Requests, RegExp, NLTK, HTML2Text, BeautifulSoup, Pymorphy 2, NumPy,
Scikit-learn, Gensim, PyMystem, Pandas, TQDM.

Метод 1

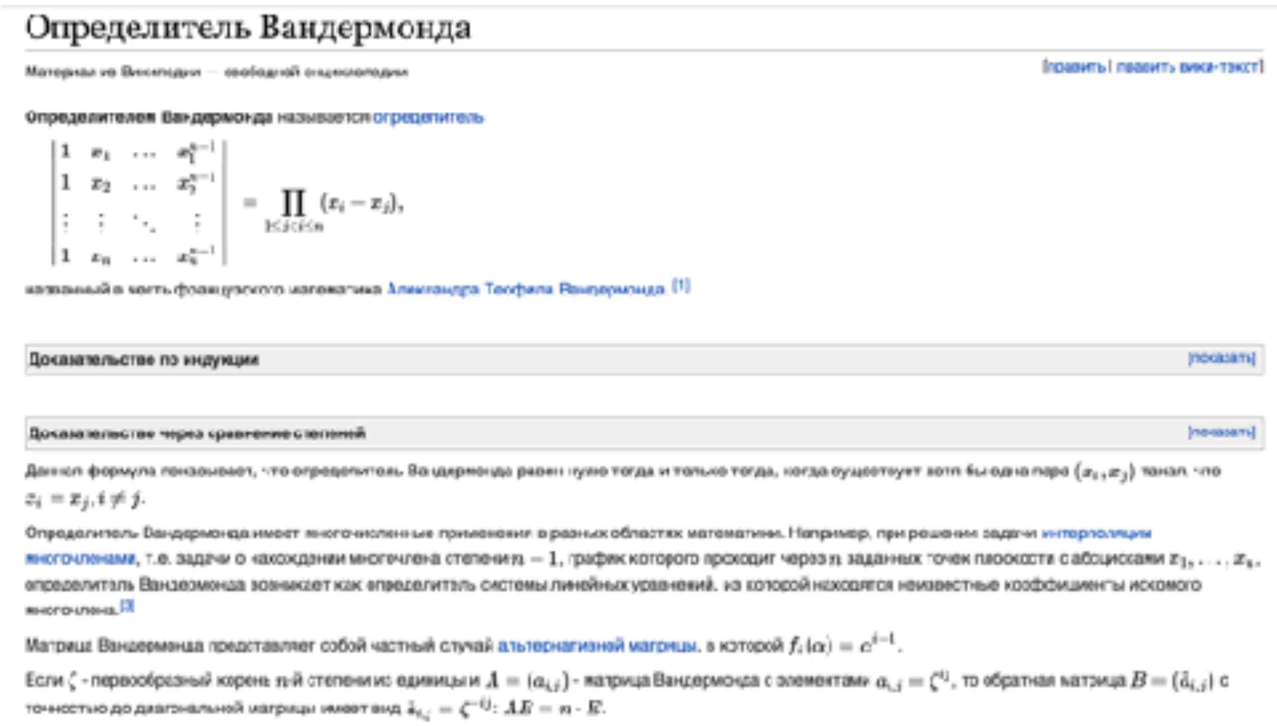
Базовый

Первый метод основан на явном нахождении лексем из названий тем в данной web-странице. Будем парсить страничку, оставляя только слова и глаголы. Далее будем искать в них слова из представленных тематик.

Пример:

1) Рассмотрим страницу

https://ru.wikipedia.org/wiki/Определитель_Вандермонда



Определитель Вандермонда

Материал из Википедии — свободной энциклопедии

[править | править вики-текст]

Определителем Вандермонда называется определитель

$$\begin{vmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ 1 & x_2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^{n-1} \end{vmatrix} = \prod_{1\le j<i\le n} (x_i - x_j),$$

названный в честь французского математика Александра Теофила Вандермонда.[1]

Доказательство по индукции [показать]

Доказательство через сравнение степеней [показать]

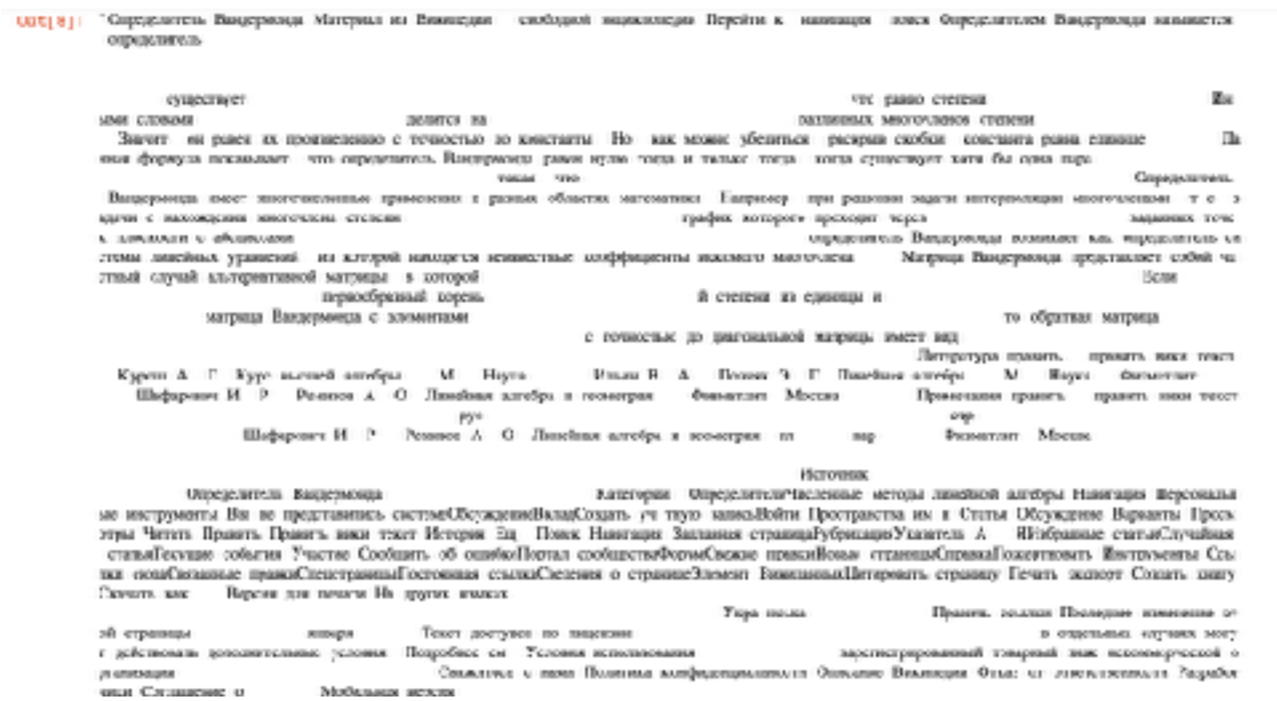
Данная формула показывает, что определитель Вандермонда равен нулю тогда и только тогда, когда существует хотя бы одна пара (x_i, x_j) такая, что $x_i = x_j, i \neq j$.

Определитель Вандермонда имеет многочисленные применения в разных областях математики. Например, при решении задачи интерполяции многочленами, т.е. задачи о нахождении многочлена степени $n-1$, график которого проходит через n заданных точек плоскости с абсциссами $x_1, \dots, x_n$, определитель Вандермонда возникает как определитель системы линейных уравнений, из которой находятся неизвестные коэффициенты искомого многочлена.[2]

Матрица Вандермонда представляет собой частный случай альтернантной матрицы, в которой $f_i(\alpha) = \alpha^{i-1}$.

Если ζ - первообразный корень n-й степени из единицы и $A = (a_{i,j})$ - матрица Вандермонда с элементами $a_{i,j} = \zeta^{ij}$, то обратная матрица $B = (b_{i,j})$ с точностью до диагональной матрицы имеет вид $b_{i,j} = \zeta^{-ij}$: $AB = n \cdot E$.

2) Путем нехитрых взаимодействий получаем русский текст:

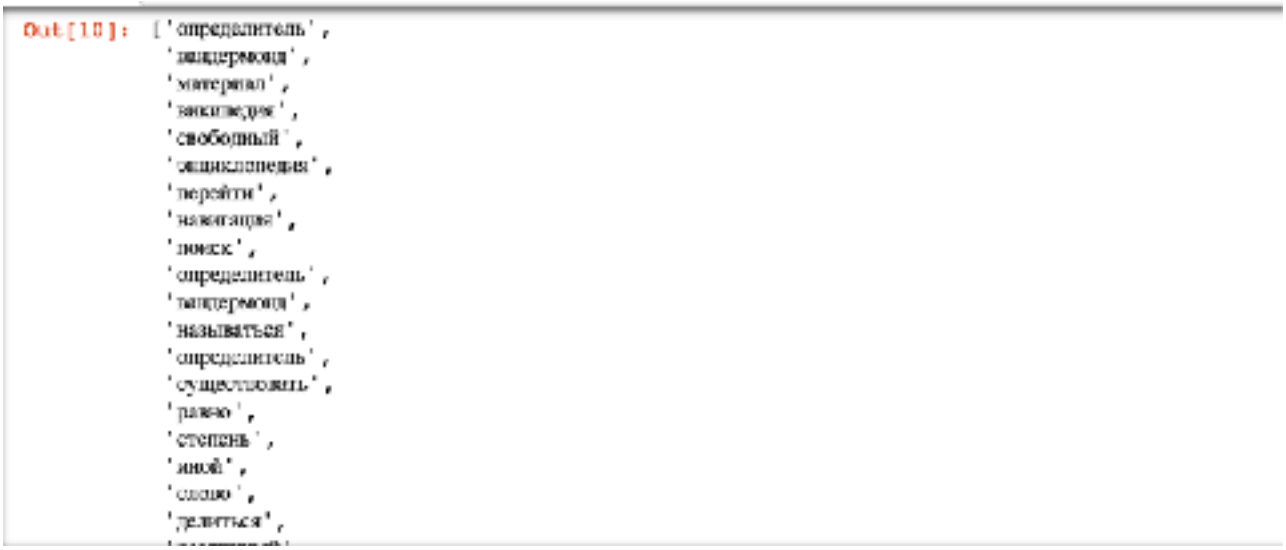


3) Выделяем среди этих слов такие слова, что:

- их длина больше 2 и меньше 16
- это существительное или глагол
- оно не входит в список стоп-слов

(Все эти шаги подробно проиллюстрированы в файле WebClassifier-1.ipynb)

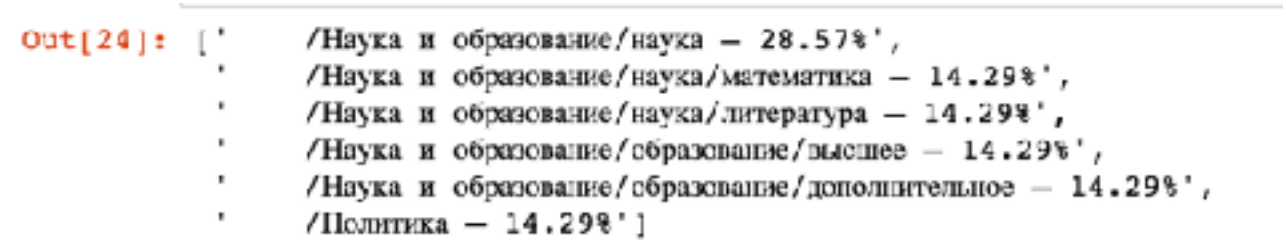
4) Теперь, преобразуя все эти слова в начальную форму получаем список слов:



5) Используя Bag of words модель векторизации слов (каждое слово представляется вектором, где каждый скаляр вектора представляет собой количество вхождений слова в выборку) и TF-IDF методы (частота слова умножить на обратную встречаемость в документах) определяем соответствующие значения каждого слова.

6) И в заключении, осталось найти каждый из полученных ответов в названиях тем. Для этого нужно представить эти слова в нормальной форме. Создается вспомогательный массив, чтобы по нему восстановить изначальную тему.
К примеру:
‘Внутренний' -> ' **/Политика/Внутренняя**’
‘дополнительный': -> ‘ **/Наука и образование/образование/дополнительное**’

Соберем все вместе, получим ответ:



Плюсы:

Работает быстро

Не требует много знаний

Минусы:

Невозможно определить точность (но, скорее всего, она довольно низкая :))

Слова в тексте не всегда определяют его тематику

Метод 2

Word2Vec

Второй метод основан на первом и отличается от него подключениям обученной модели Word2Vec и поиском слов со страницы не сразу в названиях тематик, а в словаре модели, и лишь результаты конечного вектора ищутся в словах начальных тематик.

Предобученная модель (1) взята отсюда:
http://ling.go.mail.ru/misc/dialogue_2015.html#rnc
Предобученная модель (2) взята отсюда:
http://ling.go.mail.ru/misc/dialogue_2015.html#news

Пример:

Повторим действия 1-5 из прошлой модели, получим набор слов и TF-IDF значений.

Теперь, используя обученную модель 1 и 2 проделаем следующие действия:

1. Найдем начальную форму из темы в словаре модели
2. Умножим этот вектор на значение TF-IDF слова
3. Проделаем так со всеми найденными в словаре модели Word2Vec ключами и в итоге получим результирующий вектор - сумму всех векторов.

Магия Word2Vec позволяет найти похожие на вектор слова из словаря, нам хватит сотни:

```
In [79]: hundred_similar_words
```

Last executed 2018-10-19 20:47:12 in 9ms

```
Out[79]: [('наука', 0.8318927884101868),
 ('математика', 0.6977936029434204),
 ('литература', 0.6834399700164795),
 ('физика', 0.6183444857597351),
 ('биология', 0.6048916578292847),
 ('естествознание', 0.5761309862136841),
 ('физико-математический', 0.5751566290855408),
 ('информатика', 0.5618166923522949),
 ('лингвистика', 0.5615713596343994),
 ('языкознание', 0.5526806116104126),
 ('культурология', 0.5335828065872192),
 ('филологический', 0.5316666960716248),
 ('обществознание', 0.531338632106781),
 ('химия', 0.5215914249420166),
 ('кибернетика', 0.5192195177078247),
 ('естественнонаучный', 0.517939031124115),
 ('обществоведение', 0.5163986682891846),
 ('литературоведение', 0.5140202641487122),
 ('член-корреспондент', 0.5103638172149658),
 ('астрономия', 0.5091902017593384),
 ('математик', 0.5085898041725159),
 ('алгебра', 0.501548707485199),
 ('педагогика', 0.49928444623947144),
 ('естественно-научный', 0.4954244494438171),
 ('доцент', 0.4946599304676056),
 ('география', 0.4937931299209595),
 ('профессор', 0.49146321415901184),
```

Можем снова проделать процедуру восстановления исходных тем по этим словам, умножим эти числа на 100 и получим ответ:

1) Используя модель News

```
Out[62]: ['     /Наука и образование/наука — 83.19%',
          '     /Наука и образование/наука/математика — 69.78%',
          '     /Наука и образование/наука/литература — 68.34%',
          '     /Наука и образование/наука/физика — 61.83%']
```

2) Используя модель Russian National Library

```
Out[7]: ['         /Наука и образование/наука — 90.71%',
         '         /Наука и образование/наука/математика — 68.50%',
         '         /Наука и образование/наука/литература — 67.38%']
```

Плюсы:

Более высокая точность

Минусы:

Работает немного дольше

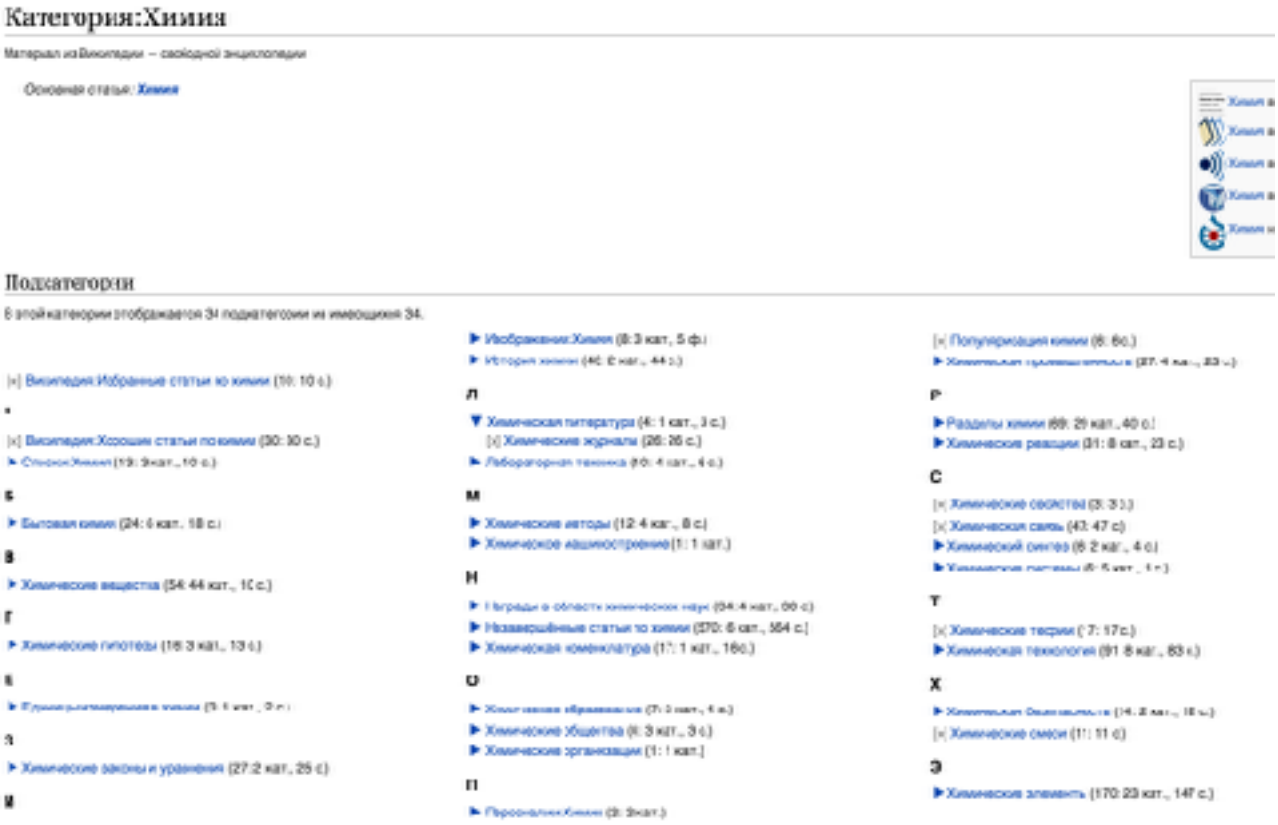
Требуется обучить модель / найти готовую

Не ищет специфичные тематики (GoTo, E-Contenta) - не показывает совсем ничего

Метод 3

Линейные модели

Пришлось сначала проделать большую работу, чтобы найти русский датасет для обучения. Ушла неделя, чтобы понять, что всё придется делать ручками. Изначально хотелось выкачивать нужные странички википедии, но с категориями на вики все слишком мутно:



Первым делом, нашёлся сайт http://www.dmoz.org/ с миллионом категорий и 3.9 млн сайтов. http://rdf.dmoz.org/ - Отсюда можно скачать все ссылки и их категории. А отсюда https://github.com/gr33ndata/dmoz-urlclassifier/blob/master/dmoz2csv.py я получил распаршенный csv файл.

Теперь следовало найти все русские темы в этом файле (и немного преобразовать их, а то некоторые склеились)

```
In [6]: raw_table.Тема.value_counts()
```

Last executed 2016-10-20 17:48:44 in 24ms

Out[6]:

Бизнес_и_экономика	3095
Новости_и_СМИ	1743
Торговля	1357
Путешествия_и_туризм	1044
Образование	1018
Строительство_и_эксплуатация	981
Здоровье	863
Государство	825
Недвижимость	731
Продукты_питания	695
Отдых_и_спорт	693
Компьютеры_и_интернет	674
Общество_и_культура	672
Автомобили	656
Искусство_и_развлечения	640
Мебель	577

Отсюда я выбрал эти темы: *Бизнес и экономика*, *Новости и СМИ*, *Торговля*, *Путешествия и туризм*, *Образование*, *Здоровье*, *Компьютеры и интернет*, *Химия*, *Кино*, *Фитнес*, *Высшее образование*, *Литература*.

Скачал по 100 сайтов (а точнее их главных страниц) и провернул уже знакомую операцию отбора существительных и глаголов в начальной форме. Даже на этом этапе приходилось много раз перескачивать. То ссылки кривые попадались (и вместо 100 наборов слов получалось 0), то данные сериализовались как-то неправильно. Но в итоге я смог собрать данные для обучения.

Были выбраны две модели: SGD Classifier и RandomForestClassifier.

Далее уже знакомая векторизация этих слов, небольшая обработка данных. Pipeline и первые численные результаты. Точность - 79.5% на первой модели и 62% на второй. Думаю, можно сделать лучше. Так что я оставил только первую модель и запустил Grid Search, получил конфигурацию на 75%. Тренировочные данные берутся произвольно, поэтому результаты разнятся и я решил, что эту ситуацию можно исправить только в корне.

```
Best score: 0.751
Best parameters set:
        classifier__alpha: 0.001
        classifier__loss: 'log'
        vect__max_df: 0.5
        vect__max_features: 5000
        vect__ngram_range: (1, 2)
```

Я подумал брать не по 100, а по 250 сайтов, при этом не разбивая страницу на слова и приводя их в нормальную форму, а брать как есть, просто проходить RegExpTokenizer'ом и делать стемминг.

Раньше:

Сейчас:

Прошло 2 часа, как закачался новый датасет.
Я обучил модель с GridSearch и понял, что предлоги не нужны.

```
text_clf.fit(X_train, Y_train)
predicted = text_clf.predict(X_test)
np.mean(predicted == Y_test)
```

```
Pipeline(steps=[('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=0.5, max_features=None, min_df=1,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip...    penalty='l2', power_t=0.5, random_state=None, shuffle=True,
       verbose=0, warm_start=False))])

0.7165532879818594
```

Я убрал их и что-то не пошло :))

```
Out[64]: Pipeline(steps=[('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=0.5, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip...    penalty='l2', power_t=0.5, random_state=None, shuffle=True,
               verbose=0, warm_start=False))])

Out[64]: 0.7210984359414968
```

По крайней мере, я убежден, что всё-таки теперь мой датасет более качественный. Слова в начальной форме без предлогов и не обрезанные. MyStem отработал хорошо.

Под конец возникли самые дурацкие проблемы. Кол-во фичей не совпадало, вылезали странные ошибки «Lower not found», «max_df < min_df» и еще куча других. Я рад, что исправил их, хоть и миллионом костылей.
Весь код есть в файлах WebClassifier(_,-1,-2,-3).ipynb, Preparing Data for WebClassifier-3.ipynb
Конечные итоги:

```
In [159]: method3("https://ru.wikipedia.org/wiki/Определитель_Вандермонда")
          Last executed 2016-10-22 21:02:40 in 470ms
Out[159]: '      /Наука и образование/образование'

In [160]: method3("http://goto.msk.ru/hackathon/")
          Last executed 2016-10-22 21:02:41 in 270ms
Out[160]: '      /Наука и образование/образование'

In [161]: method3("https://www.kinopoisk.ru/film/648449/")
          Last executed 2016-10-22 21:02:42 in 2.01s
Out[161]: 'Кино'

In [162]: method3("https://e-contenta.com/ru/")
          Last executed 2016-10-22 21:02:46 in 555ms
Out[162]: 'Компьютеры_и_интернет'

In [163]: method3("https://ru.wikipedia.org/wiki/C%2D%2D")
          Last executed 2016-10-22 21:02:47 in 105ms
          Возможно, вы предоставили неправильную ссылку
```

Перспективы развития

- Написать четвертый метод - с использованием нейронных сетей
- Найти более точную выборку данных
- Развивать свои Machine Learning Skills и делать всё лучше
- ~~Попасть на GoTo School~~