Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра Математических методов прогнозирования

Курсовая работа

Методы распознавания сарказма в тексте

Выполнил:
студент 517 группы
Кибитова Валерия Николаевна

Научный руководитель:
д.ф-м.н., профессор
Дьяконов Александр Геннадьевич

Москва, 2016

Содержание

1 Введение

Сарказм – это способ выражения мыслей таким образом, что предполагаемое и буквальное значение текста становятся противоположными. Сарказм часто используется, чтобы выразить негативное сообщение, используя слова позитивной окраски. Например: "Как же я люблю получать спам!". Автоматическое обнаружение сарказма является важной задачей в области распознавания настроения, так как саркастичная фраза, содержащая позитивные слова, но в целом передающая негативное сообщение может быть неправильно распознана автоматической системой распознавания настроения.

В настоящее время существует ряд исследований посвященных автоматическому распознаванию сарказма в тексте. Все они подразделяются на два основных направления: алгоритмы распознавания сарказма, основанные на конструировании признакового пространства и применении методов машинного обучения и алгоритмы, основанные на лингвистической структуре сарказма. Однако было замечено, что в настоящее время для применения распознавания сарказма, авторами не были исследованы признаки, основанные на применении моделей word2vec.

В данной работе будет вестись повествование о различных методах, которые могут использоваться при распознавании сарказма в тексте на примерах конкретных представителей этих методов. Также будет предложен новый метод, который может быть использован при распознавании сарказма и будет проведено исследование, посвященное композициям различных алгоритмов.

2 Постановка задачи

Задача распознавания сарказма состоит в следующем: пусть T – это множество текстов, необходимо определить отображение: $T \to \{0, 1\}$, где класс 1 – это класс текстов, содержащий сарказм, и класс 0 – класс текстов, в которых сарказм отсутствует.

Задача данной работы состоит в исследовании существующих методов распознавания сарказма, а также в построении новых методов, которые позволят улучшить качество классификации.

Цель данной работы заключается в определении методов, которые показывают наилучшее качество при распознавании сарказма.

3 Распознавание сарказма при помощи алгоритма, основанного на парсинге текста

Данный алгоритм [1] относится к классу алгоритмов, основанных на лингвистической структуре сарказма и полагается на следующие факты. Текст содержит сарказм, если в предложении содержится контраст положительной оценки

и негативной ситуации или негативной ситуации и положительной оценки. Кроме того, авторами данного алгоритма было отмечено, что если текст начинается с междометия, после которого следует прилагательное или наречие, то данный текст также является саркастичным. Например: "Oh wow look at the most realistic doughnuts in a video game (Ух ты, посмотри на самые реалистичные пончики в видеоигре)".

На основании данных фактов весь алгоритм состоит из двух частей:

- Идентификация сарказма основанная на парсинге. Распознает сарказм в случаях сочетания положительной оценки и негативной ситуации и негативной оценки и положительной ситуации (PBLGA – Parsing Based Lexical Generation Algorithm).

- Идентификация сарказма в текстах, начинающихся с междометий (IWS – Interjection Word Start).

В данном алгоритме под парсингом понимается процесс анализа грамматической структуры предложения. Подавая парсеру на вход последовательность слов, парсер определяет связи между частями речи в предложении и устанавливает связи между ними. В качестве парсера при реализации данного алгоритма использовалась библиотека TextBlob[1], также данная библиотека использовалась для определения эмоциональной окраски слов. Парсер, который реализован в библиотеке $TextBlob$ при выделении грамматической структуры текста позволяет выделять следующие типы фраз: фраза-существительное, фраза-глагол, фраза-наречие, фраза-прилагательное.

Введем следующие обозначения: P_{verb}: фраза-глагол, P_{noun}: фраза-существительное, $P_{adjective}$: фраза-прилагательное, P_{adverb}: фраза-наречие, $_{gr}$: коллекция, содержащая грамматические структуры текстов, T_{gr}: грамматическая структура текста, $S_{sentiment}$: множество фраз, описывающих эмоции, $S_{situation}$: множество фраз, описывающих ситуации, $S_{p\_sentiment}$: множество фраз, описывающих положительные эмоции, $S_{n\_sentiment}$: множество фраз, описывающих негативные эмоции, $S_{p\_situation}$: множество фраз, описывающих положительные ситуации, $S_{n\_situation}$: множество фраз, описывающих негативные ситуации, T_{score}: эмоциональная оценка текста, P: фраза, T: текст, C: коллекция текстов.

В начале работы данного алгоритма осуществляется парсинг для каждого из текстов. Затем выделяются части исходного текста(find_subset). В зависимости от того какие типы фраз встречаются в каждой части исходного текста, она определяется как фраза, описывающая эмоции или ситуации . После этого для каждой ситуации и эмоции находится оценка эмоциональной окраски(sentiment_score), которая определеяется следующим образом:

$$\text{sentiment\_score} = PR - NR, PR = \frac{PWP}{TWP}, NR = \frac{NWP}{TWP},$$

[1] `https://textblob.readthedocs.io/en/dev/`

Algorithm 1 PBLGA

$S_{situation} = \emptyset, S_{sentiment} = \emptyset, S_{p\_sentiment} = \emptyset, S_{n\_sentiment} = \emptyset, S_{p\_sentiment} = \emptyset, S_{n\_sentiment} = \emptyset$
for T in C **do**
    $k = \text{find\_parse}(T)$
    $_{gr} = _{gr} \cup k$
end for
for T_{gr} in $_{gr}$ **do**
    $k = \text{find\_subset}(T_{gr})$
    if $k == P_{noun}||P_{adverb}||(P_{noun} + P_{verb})$ **then**
        $S_{emotion} = S_{emotion} \cup k$
    else if $k == P_{verb}||(P_{adjective} + P_{verb})||(P_{verb} + P_{adverb})||(P_{adjective} + P_{verb})||(P_{verb} + P_{noun})||(P_{verb} + P_{adverb} + P_{adjective})||(P_{verb} + P_{adjective} + P_{noun})||(P_{adverb} + P_{adjective} + P_{noun})$ **then**
        $S_{situation} = S_{situation} \cup k$
    end if
end for
for P in $S_{sentiment}$ **do**
    $T_{score} = \text{sentiment\_score}(P)$
    if $T_{score} > 0.0$ **then**
        $S_{p\_sentiment} = S_{p\_sentiment} \cup P$
    else if $T_{score} < 0.0$ **then**
        $S_{n\_sentiment} = S_{n\_sentiment} \cup P$
    else
        Neutral Sentiment Phrase
    end if
end for
for P in $S_{situation}$ **do**
    $T_{score} = \text{sentiment\_score}(P)$
    if $T_{score} > 0.0$ **then**
        $S_{p\_situation} = S_{p\_situation} \cup P$
    else if $T_{score} < 0.0$ **then**
        $S_{n\_situation} = S_{n\_situation} \cup P$
    else
        Neutral Situation Phrase
    end if
end for

где PWP – число положительных слов в данной фразе, NWP – число негативных слов в данной фразе, TWP – общее число слов в данной фразе. В зависимости от sentiment_score определяется в какой список положительных или негативных эмоций (ситуаций) заносится данная фраза. После построения списков положительных и негативных эмоций, а также положительных и негативных ситуаций осуществляются предсказания для текстов. Если в тексте одновременно присутствует фраза, описывающая негативную ситуацию и фраза, которая описывает положительные эмоции, или фраза, описывающая положительную ситуацию вместе с фразой описывающей негативные эмоции, то текст распознается как содержащий сарказм. Данный алгоритм, используется в том случае, если текст не содержит междометий.

Алгоритм IWS используется для распознавания сарказма в текстах, которые начинаются с междометий. Например таких как: wow, yay, yeah, aha, oh.

При реализации данного алгоритма для определения тегов частей речи текста использовалась библиотека TextBlob.

Обозначения: ADJ: прилагательное, V: глагол, ADV: наречие, N: существительное, IN: междометие, T: текст, C: коллекция текстов, $_{tags}$ – тексты, содержащее список тегов для каждого текста, tag: тег, T_{tags}: теги текста, tag_1: первый тег, tag_{in}:непосредственный следующий тег , tag_n: следующий тег.

Algorithm 2 IWS

for T in C **do**
    $k = \text{find\_postag}(T)$
    $S_{tags} = S_{tags} \cup k$
end for
for T_tags in S_{tags} **do**
    $t = \text{find\_subset}(T_{tags})$
    $tag_1 = \text{find\_first\_tag}(t)$
    $tag_{in} = \text{find\_immediate\_next\_tag}(t)$
    $t_n = \text{find\_next\_tag}(t)$
    if $(tag_1 == IN)\&\&(tag_{in} == (ADJ||ADV))$ **then**
        Tweet is sarcastic
    else if $(tag_1 == IN)\&\&(tag_n == (ADV + ADJ))||(ADJ + N)||(ADV + V))$ **then**
        Tweet is sarcastic
    else if $tag_1 \neq IN$ **then**
        Invalid tweet.
    else
        Tweet is not sarcastic
    end if
end for

В начале работы алгоритма IWS для каждого текста определяются теги частей речи, которые в него входят. Затем выделяется часть текста, первым тегом

которой является междометие. Если следующие за ней теги – это теги, обозначающие глагол и прилагательное или прилагательного и существительное, или наречие и глагол, то текст классифицируется как содержащий сарказм. Рассмотрим пример: "Wow, what an amazing night this has turned out to be". В данном случае после междометия (Wow) следуют прилагательное и существительное (amazing night), следовательно текст определяется как содержащий сарказм. Если текст не содержит междометия, то тогда текст не содержит информации, необходимой для распознавания данным алгоритмом и, как следствие, не распознается. В других случаях считается, что текст не содержит сарказма.

4 Признаковые пространства, используемые для распознавания сарказма в тексте при помощи методов машинного обучения

4.1 Признаковое пространство, основанное на структуре предложений

При формировании данного признакового пространства основной целью авторов было избегать использования слов или паттернов слов как признаков. Признаковое пространство было сформировано только основываясь на структуре предложения [2].

Для формирования данного признакового пространства использовался корпус ANC(American National Corpus)[2], содержащий коллекцию из 15 миллионов слов американского варианта английского языка, вместе с частотами их употребления. Данный корпус также содержит отдельные частоты употребления слов для письменного и устного стиля речи. Другим средством, необходимым для формирования данного признакового пространства является WordNet [3], который позволяет получить для каждого слова синонимический ряд, который с ним связан (или несколько таких рядов, если слово является многозначным). Также использовался SentiWordNet [4] для того, чтобы оценить эмоциональную окраску текста (позитивность или негативность). SentiWordNet предоставляет оценку для каждого синонимического ряда в пределах от $[-1, 1]$. Оценка -1 указывает на то, что данное понятие является крайне негативным, а оценка 1 указывает на то, что понятие является позитивным. Для того чтобы получить оценки интенсивности для прилагательных и наречий, использовался набор данных[3], который содержит оценки интенсивности для различных прилагательных и наречий. Если оценка положительная, то это показывает что слово является позитивным, если отрицательная – негативным. Модуль оценки зависит от интенсивности прилагательного и наречия. Пример: $horrible(-1.9) \rightarrow bad(-1.1) \rightarrow good(0.2) \rightarrow nice(0.3) \rightarrow great(0.8)$.

[2] `http://www.anc.org/`

[3] `http://web.stanford.edu/~cgpotts/data/wordnetscales/`

Все признаковое пространство состоит из 7 групп признаков:

- Признаки, связанные с частотой. Авторы данного признакового пространства утверждают, что некоторая неожиданность в тексте может указывать на наличие в тексте сарказма. Одним из индикаторов неожиданности является встречаемость в тексте одновременно как слов с высокой частотой использования, так и слов с низкой частотой. К данной группе принадлежат 3 признака: средняя частота слов в тексте, частота самого редкого слова, разница первых двух признаков.

- Признаки, связанные со стилем написания. Так как на наличие сарказма в предложении может указывать стиль (письменный или устный), в котором данное предложение написано, были включены следующие признаки: средняя частота слов, написанных в письменном стиле, средняя частота слов, написанных в устном стиле, разница первых двух признаков.

- Признаки, связанные со структурой предложения. Для того, чтобы выделить различия в структуре предложения между текстом, содержащим сарказм и текстом без него, были выделены следующие признаки: число символов, из которых состоит текст; число слов в тексте; средняя длина слов в тексте; число глаголов, существительных, прилагательных и наречий; доля глаголов, существительных, наречий и прилагательных в тексте; число всех пунктуационных символов в тексте(точка, запятая, вопросительный и восклицательный знак); отдельное количество точек, запятых, восклицательных и вопросительных знаков в тексте; наличие слов, обозначающих смех($hahah$, lol, $rofl$, $lmao$); число смайликов (:), ;), : D, : ().

- Признаки, связанные с эмоциональной окраской предложения. Данная группа признаков создавалась для того, чтобы выявить разницу между эмоциональной окраской предложения саркастичных и не саркастичных текстов, а также выявить несбалансированность эмоций внутри предложения. Для каждого слова в тексте выявлялась его оценка с помощью SentiWordNet. К данной группе принадлежат следующие признаки: сумма всех положительных оценок; сумма всех отрицательных оценок; разность между предыдущими двумя признаками; разность между максимальной положительной оценкой и средней, разность между минимальной негативной оценкой и средней.

- Признаки, связанные с неоднозначностью. Одним из аспектов сарказма является неоднозначность высказывания, которое его содержит. Существует предположение, что если слово имеет много значений, то вероятность того, что буквальный и намеренный смысл предложения будут противоположны, увеличится. К данной группе относятся следующие признаки: среднее число значений слов в тексте; максимальное число значений слова в тексте; разность предыдущих двух признаков

- Признаки, связанные с интенсивностью наречий и прилагательных: суммарная интенсивность прилагательных(наречий), средняя интенсивность, максимальная интенсивность, разность между максимальной и средней интенсивностью

- Признаки, связанные с синонимами. Так как предложение, содержащее сарказм одновременно содержит в себе два смысла(буквальный и предполагаемый), то выбор синонима является важным при конструировании сарказма. Для каждого слова в тексте, был получен его список его синонимов, для каждого из которых была получена его частота. Для того, чтобы выделить признаки в данной группе, необходимо ввести некоторые величины. Введем следующие обозначения: syn_i – это синоним слова w_i, $f(x)$ – это частота слова x, $mean\{M\}$ – среднее значение множества M. $abs(x)$ – абсолютное значение величины x. Введем следующие величины:

  $$sl_{w_i} = |syn_i : f(syn_i) < f(w_i)|$$

  – число синонимов слова, с частотой меньшей, чем у самого слова,

  $$sg_{w_i} = |syn_{w_i} : f(syn_{w_i}) > f(w_i)|$$

  – число синонимов слова, с частотой большей чем, у самого слова,

  $$wsl_t = max_{w_i}\{|syn_i : f(syn_i) < f(w_i)|\}$$

  – величина, определяющая максимальное количество синонимов, которые могут быть у слова, с частотой меньшей частоты самого слова,

  $$wsg_t = max_{w_i}\{|syn_i : f(syn_i) > f(w_i)|\}$$

  – величина, определяющая максимальное количество синонимов, которые могут быть у слова, с частотой большей частоты самого слова.

  На основе этих величин определим следующие признаки: $mean\{sl_{w_i}\}$, $mean\{sg_{w_i}\}$, $abs(wls_t - mean\{sl_{w_i}\})$, $abs(wgs_t - mean\{sg_{w_i}\})$.

4.2 Признаковое пространство, основанное на причинах проявления сарказма в тексте.

Данное признаковое пространство строилось исходя из факторов, которые влияют на причины появления сарказма в тексте [5]. Для формирования данного признакового пространства использовались оценки эмоциональной окраски слов, из набора данных SentiStrength[4] [9]. Этот набор данных содержит оценку для каждого слова в пределах от $[-5, 5]$, где оценка 5 указывает на высокую позитивность слова, а -5 указывает на высокую негативность слова. Другой набор

[4] `http://sentistrength.wlv.ac.uk/SentStrength_Data/`

данных, который был необходим при формировании данного признакового пространства[5] содержит оценки эмоционального воздействия по шкале от $[1, 9]$ для английских слов. Где оценка 1 показывает, что слово является наименее приятным. Например: оценка для hate (ненавидеть) – 1.96, оценка для spam (спам) – 3.1, а оценка для слова joy (радость) – 8.2.

Признаки, которые были сформированы при этом подходе разделены на группы, в зависимости от того, какая причина могла вызвать генерацию сарказма.

Были выделены следующие группы признаков:

- Контраст эмоций в тексте. Одним из наиболее общих способов генерации сарказма является использование словосочетаний с контрастными эмоциональными оценками. Например: *I love getting spam emails*! (Мне нравится получать спам!). В данном случае отрицательно эмоционально окрашенное слово спам является контрастным по отношению к положительно эмоционально окрашенному слову нравится. Введем следующие обозначения: affect(w) – оценка эмоционального воздействия слова w, sentiment(w) – оценка эмоциональной окраски слова w. Для получения признаков, сформируем следующие множества:

  $$A = \{\text{affect}(w) | w \in t\} \quad S = \{\text{sentiment}(w) | w \in t\}$$

  На основе данных множеств были сформированы следующие признаки:

  $$\Delta\text{affect} = \max(A) - min(A) \quad \Delta\text{sentiment} = \max(S) - \min(S)$$

- Сарказм как сложная форма самовыражения. Как широко известно, выражения, которые содержат сарказм являются как правило трудно доступными для понимания. Так как одним из факторов влияющих на восприятие текста является наличие в тексте длинных слов, было сформировано следующие распределение: $L = \{l_i\}$, где l_i отвечает за число слов длины i в тексте. Для того, чтобы описать данное распределение использовались следующие 6 признаков:

  $$< \text{E}[l_w], \text{med}[l_w], \text{mode}[l_w], \sigma[l_w], \min l_w, \max l_w >,$$

  где E$[l_w]$ – среднее распределения, med$[l_w]$ – медиана распределения, mode$[l_w]$ – мода распределения, $\sigma[l_w]$ – стандартное отклонение распределения, $\max l_w$ – максимальное значение в распределении, $\min l_w$ – минимальное значение в распределении.

- Сарказм как средство выражения эмоций. Так как выражения, которые содержат сарказм, являются довольно эмоциональными, были сформированы следующие признаки, которые описывают эмоциональную окраску текста.

[5] `http://crr.ugent.be/archives/1003`

Было сформировано распределение $SD = \{s_i\}$, – где s_i – это число слов, имеющих оценку эмоциональной окраски i. SD состоит из 11 значений, все эти 11 значений были выбраны как признаки, также как признаки использовалось 6-признаковое представление для данного распределения:

$$< \mathrm{E}[sd_w], \mathrm{med}[sd_w], \mathrm{mode}[sd_w], \sigma[sd_w], \min sd_w, \max sd_w > .$$

Аналогично было составлено распределение оценок эмоционального воздействия слов в тексте: $AD = \{a_i\}$, где a_i – это число слов, которые имеют оценку эмоционального воздействия i. Также были включены следующие признаки: число слов, которые содержатся в данных с оценками эмоционального воздействия; число слов, с ненулевыми оценками эмоциональной окраски; а также оценку эмоциональной окраски всего твита, которая была получена с помощью $TextBlob$. Так как одним из способов выражения эмоций является присутствие в тексте бранных слов, то данная черта была представлена булевым признаком. Список бранных слов был взят из списка бранных английский слов[6].

- Сарказм как мера знакомства с языком. Существует предположение, что люди, которые используют сарказм, должны обладать хорошими знаниями языка. Чтобы сформировать представление о грамматических знаниях пользователя было сформировано распределение частей речи слов в тексте. Для определения частей речи в предложении использовался определитель частей речи из $TextBlob$. Так как $TextBlob$ умеет определять 34 части речи. То в признаковое пространство были включены 34 дополнительных признака.

- Сарказм как форма письменного самовыражения. Известно, что текст, содержащий сарказм, часто имеет особый стиль написания. Поэтому, чтобы сформировать представление о структуре предложений, содержащих сарказм использовались следующие признаки: присутствие повторяющихся символов (3 или больше) во всех словах и в словах, выражающих эмоции, число символов, число различных символов, число слов с большой буквы, число существительных, глаголов, прилагательных и наречий, используемых в тексте, число стоп-слов в тексте, лексическая плотность текста (отношения числа прилагательных и наречий к общему числу слов в тексте), число слов усилителей (список слов-усилителей был взят из данных в SentiStrength). Было сформировано распределение пунктуации в твите, состоящее из 6 значений: ".!,"*'".

- Сарказм как особая структура предложения. Предполагается, что предложения, содержащие сарказм, отличаются по своей структуре от предложений, которые его не содержат. На основе этого предположения были сформированы следующие признаки: теги частей речи первых трех слов в тексте;

[6] `http://www.noswearing.com/dictionary`

позиция первого слова, имеющего ненулевую оценку эмоциональной окраски в тексте; позиция первого слова, содержащегося в данных с оценкам эмоционального воздействия.

4.3 Признаковое пространство, основанное на контрасте эмоций и согласованности текста.

Основная идея, которая были использована авторами при формировании данного признакового пространства – это определить противоречие эмоций в тексте, при этом определив его согласованность. Данный подход основан на том, что один из признаков наличия сарказма в предложении – это контраст различных эмоциональных окрасок в логически согласованном тексте [6]. До начала формирования основного признакового пространства, на котором в последствии будет обучаться алгоритм машинного обучения необходимо вычислить некоторые характеристики текста, а именно: сумму положительных оценок слов, сумму отрицательных оценок слов, а также общую согласованность текста.

Для определении оценки для каждого отдельного слова использовались TextBlob и SentiStrength. TextBlob предоставляет оценку для каждого слова в пределах от $[-1, 1]$. Для того, чтобы учесть обе оценки, оценки, которые были получены с помощью TextBlob, умножались на 5.

Введем следующие обозначения: SS - SentiStrength, TB – TextBlob, pos_w – слово, обладающее положительной оценкой, T – рассматриваемый текст, neg_w – слово, обладающее отрицательной оценкой, получим, что оценка для слова (w_score), сумма положительных оценок (sum_pos_score), сумма отрицательных оценок оценок (sum_neg_score) определяются следующим образом:

$$\text{w\_score}(w) = \begin{cases} \text{polarity\_score}(w), & \text{if } w \in TB \text{ or } SS \\ \text{average\_polarity\_score}(w), & \text{if } w \in SS \text{ and } TB \end{cases}$$

$$\text{sum\_pos\_score} = \sum_{pos\_w \in T} \text{w\_score}(pos\_w)$$

$$\text{sum\_neg\_score} = \sum_{neg\_w \in T} \text{w\_score}(neg\_w)$$

Для того, чтобы проверить согласованность двух последующих предложений в тексте, авторами статьи [6] был предложен следующий метод. Предложения s_1 и s_2 согласованы, если существует такое слово w_1 в предложении s_1 и слово w_2 в предложении s_2, что выполняется одно из условий:

- w_1 и w_2 идентичные местоимения;
- w_1 и w_2 идентичны как строки, при условии что w1 и w2 не являются стоп-словами;

- перед словом w_2 стоит артикль the;
- перед словом w_2 стоит одно из указательных местоимений: $this$, $that$, $these$, $those$.

Весь текст определяется как логически связанный, в том случае, если каждое последующее предложение логически связано с предыдущим. На основе введенных характеристик формировалось следующее признаковое пространство:

- N-граммы, где N=1, 2, 3. N граммы представляют собой последовательности слов, состоящие из 1, 2, 3 слов соответственно. Данные признаки являются бинарными и являются индикаторами того, присутствует данная последовательность слов в тексте или нет.
- Признаки, идентифицирующие, что в тексте содержится противоречие эмоциональных окрасок. Признак contra равен 1 в том случае, если текст состоит из 1 предложения и sum_pos_score $\neq 0$ и одновременно sum_neg_score $\neq 0$. Признак contra_coher равняется 1, в том случае, если текст состоит из более чем 1 предложения, является согласованным и sum_pos_score $\neq 0$ и одновременно sum_neg_score $\neq 0$.
- Признаки, определяющие эмоциональную окраску предложения:

$$pos\_low = 1,\ if\ sum\_pos\_score <= 1$$

$$\text{pos\_medium} = 1,\ \text{if } 1 < \text{sum\_pos\_score} <= 2$$

$$\text{pos\_high} = 1,\ \text{if sum\_pos\_score} > 2$$

  Аналогичным образом определяются признаки neg_low, neg_medium и neg_high.

- Признаки, учитывающие пунктуацию и специальные символы: для того, чтобы определить набор признаков, определялись следующие характеристики из которых в последствии формировались признаки: P_1 – число смайликов (список существующих смайликов был взять из данных SentiStrength; P_2 – число последовательностей, в которых пунктуационные символы повторяются, P_3 – число последовательностей, в которых буквы повторяются, P_4 – число слов, написанных большими буквами, P_5 – число сленговых слов и слов-усилителей (списки сленговых слов и слов-усилитей были взяты из набора данных SentiStrength). На основе данных характеристик определялись признаки:

$$P_i\_low = 1, \text{if } P_i == 0$$

$$P_i\_medium = 1, \text{if } 1 <= P_i <= 3$$

$$P_i\_high = 1, \text{if } P_i >= 4$$

4.4 Признаковое пространство, сформированное на основе использовании моделей word2vec

Компания Google реализовала метод, который позволяет учитывать контекст слов, при обработке текстов, в то же время сокращая размер данных [7]. Word2Vec представляет собой на самом деле два разных метода: Continuous Bag of Words (CBOW) и Skip-gram. В CBOW методе, целью алгоритма является предсказать слово, на основе слов, которые его окружают. В методе Skip-gram целью является предсказать слова, которые окружают данное слово. Оба метода используют нейронную сеть в качестве алгоритма для обучения. Вначале обучения каждое слово – это случайный n-размерный вектор. Во время обучения алгоритм изучает оптимальный вектор для каждого слова, используя метод $CBOW$ или $Skip-gram$.

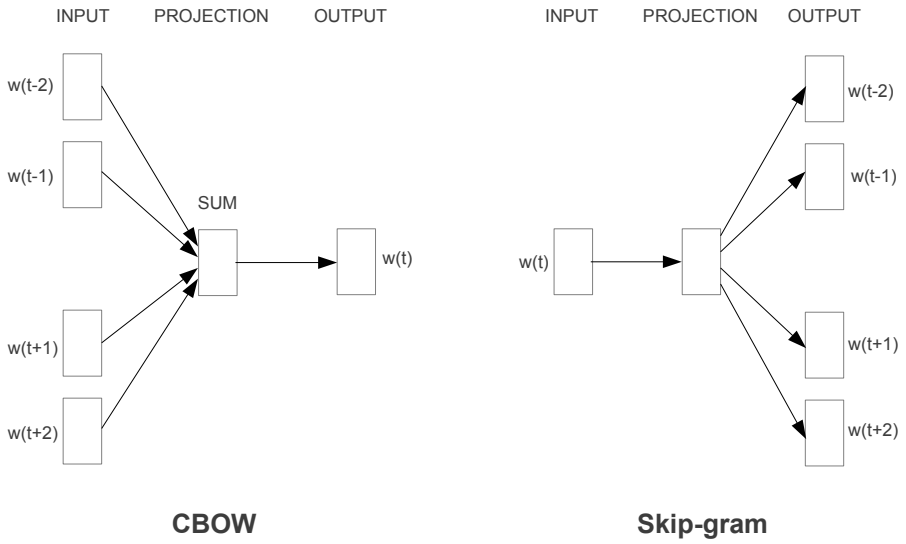

Рис. 1: Основной принцип работы методов Word2Vec

После того, как модель обучена данный метод позволяет находить слова, похожие на данное. А также измерять похожесть между двумя словами представленными векторами $\{x_i\}_{i=1}^n$ и $\{y_i\}_{i=1}^n$, используя косинусное расстояние:

$$\cos(x, y) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2}\sqrt{\sum_{i=1}^n y_i^2}}.$$

Также с помощью обученной модели существует возможность определить схожесть двух множеств слов (n_similarity)$\{a_i\}_{i=1}^k$ и $\{b_i\}_{i=1}^m$, в котором каждое слово представлено вектором размерности n, следующим образом:

$$\text{n\_similarity}(A, B) = \cos\left(\frac{\sum_{i=1}^k a_i}{k}, \frac{\sum_{i=1}^k b_i}{m}\right).$$

Данные возможности word2vec можно использовать для распознавания сарказма в тексте. В данной работе были использованы 2 предобученные модели word2vec: модель, обученная на данных Google News[7], которая содержит вектора размерности 300 для 3-миллионов слов(обозначение M_1), а также модель(обозначение модель M_2), обученная на 400 миллионах данных твиттера[8].

На основе данных моделей были сформированы следующие признаки. Так как сарказм характеризуется тем, что слова, которые не являются совместимыми по смыслу начинают находится рядом, то были вычислены косинусные меры схожести между всеми словами, находящимися на расстоянии не более 5 слов в тексте, используя модель M_1, при этом из текста были удалены все стоп-слова. Обозначим выборку, состоящую из всех таких расстояний за S. В качестве признаков использовались 6 статистик данной выборки:

$$< \mathrm{E}[S], \mathrm{med}[S], \mathrm{mod}[S], \sigma[S], \min S, \max S >$$

$\mathrm{E}[S]$ – среднее, $\mathrm{med}[S]$ – медиана, $\mathrm{mod}[S]$ – мода, $\sigma[S]$ – стандартное отклонение в в, $\min S$ – максимальное значение, $\max S$ – минимальное значение.

Для того чтобы оценить на сколько множество слов, содержащееся в тексте, похоже на сарказм использовался следующий признак: n_similarity(text_words, "sarcasm"), где text_words – это множество слов, которые содержатся в тексте.

Учитывая тот факт, что пользователь наиболее вероятно будет использовать сарказм, когда ему грустно или он злой, и наименее вероятно когда он счастлив, были определены следующие признаки: n_similarity(text_words, "angry"), n_similarity(text_words, "sad"), n_similarity(text_words, "happy").

Также в качестве признаков был использован средний вектор всех слов содержащихся в тексте, построенный при использовании модели M_2.

В качестве признака, использовался также результат работы алгоритма IWS, который позволяет определить наличие сарказма, в предложениях, которые содержат междометия.

Также использовались некоторые признаки, которые позволяют оценить структуру предложения, а также его эмоциональную окраску, описанные подробно в предыдущих разделах, а именно: Δaffect, Δsentiment, P_i_low, P_i_medium, P_i_high(где i =[1, 7], и обозначает признаки характеризующие число смайликов, число последовательностей, в которых буквы повторяются, число сленговых слов и слов усилителей, число слов, написанных большими буквами).

5 Описание используемых для проведения экспериментов данных

Для проведения экспериментов использовался набор данных, состоящий из твитов, собранных из социальной сети $Twitter$, которые были собраны при помо-

[7] `https://code.google.com/archive/p/word2vec/`

щи библиотеки Tweepy[8], являющейся оболочкой Streaming API[9]. *StreamingAPI* позволяет в онлайн-режиме скачивать данные, которые появляются в данный момент. Согласно соглашению[10] эти данные разрешено использовать для развития собственных приложений и сервисов, то есть том числе и для исследований. Для того, чтобы собрать данные, которые содержат сарказм, собирались данные, которые содержат хештеги: *#sarcasm*, *#sarc*, *#sarcastic* и *#sarcastictweet*. Для сбора данных, не содержащих сарказм собиралась случайная выборка из данных *Twitter*, также используя *Tweepy*. Сбор данных осуществлялся непрерывно в течении двух недель.

Основным недостатком сбора данных из *Twitter* является то, что данные сильно зашумлены. Твит может быть помечен как саркастический в том случае, когда сарказм содержится не в тексте твита, а в ссылке на картинку или статью, и понять что в тексте твита содержится сарказм без анализа ссылки невозможно. Некоторые твиты являются ответами на другие твиты, и не зная контекста, понять что в тексте содержится сарказм невозможно. Для устранения этих проблем из собранного набора даных устранялись твиты, которые содержали в себе ссылки или были ответами на другие твиты. Твит помечался как саркастичный, если он содержал один из следующих хештегов: *#sarcasm*, *#sarc*, *#sarcastic* и *#sarcastictweet*. Для того, чтобы исследовать только текст содержащийся в твите, из твита были удалены все хештеги и упоминания других пользователей. Если после удалений в твите содержалось более 3-х слов, то он добавлялся в окончательный набор данных. После этого, из набора данных удалялись все дубликаты. Для того чтобы оценить качество алгоритма, из этих данных генерировалась подвыборка в котором саркастичные и не саркастичные тексты находились в соотношении $1:1$. Общее количество текстов[11] составило 6784.

6 Экспериментальное исследование описанных методов

Определим величины, которые используются при описании метрик качества, используемых для оценки модели.

T_p – число истинных ответов, принадлежащих классу 1, определенных моделью.

T_n – число истинных, принадлежащих классу 0, определенных моделью.

F_p – число ложных ответов, принадлежащих классу 1, определенных моделью.

F_n – число ложных ответов, принадлежащих классу 0, определенных моделью.

$P = T_p + F_n$.

$N = T_n + F_p$.

[8] `http://www.tweepy.org`

[9] `https://dev.twitter.com/streaming/overview`

[10] `https://dev.twitter.com/overview/terms/agreement-and-policy`

[11] `https://github.com/ValeryKi/SarcasmDetection/blob/master/sarcasm_set_small.csv`

Для оценки качества модели были использованы следующие метрики:

- Точность (accuracy)= $\frac{T_p+T_n}{P+N}$
- Полнота (recall) = $\frac{TP}{P}$
- Точность (precision) = $\frac{TP}{TP+FP}$
- F-мера = $\frac{2*precision*recall}{precision+recall}$
- AUC равна вероятности того, что для случайно выбранных $x_1 \in C_1$ и $x_2 \in C_0$ будет выполнено: $p(x_1) > p(x_0)$. Где $C_{1(0)}$ – единичный и нулевые классы соответственно. $p(x)$ – вероятность принадлежности классу C_1, присвоенная классификатором.

Для оценки модели μ с помощью заданных метрик качества использовалась процедура кросс-валидации по 10 блокам, которая заключается в следующем: выборка X длины L случайным образом разбивается на 10 непересекающихся блоков одинаковой (или почти одинаковой) длины $k_1, \ldots, k_{10}$: $X^L = X_1^{k_1} \cup \cdots \cup X_{10}^{k_{10}}$, $k_1 + \cdots + k_{10} = L$. Каждый блок по очереди становится контрольной подвыборкой, при этом обучение производится по остальным 9 блокам. Качество определяется как среднее функционала качества Q на контрольной подвыборке:

$$CV(\mu, X^L) = \frac{1}{q} \sum_{n=1}^{q} Q\big(\mu(X^L \setminus X_n^{k_n}), X_n^{k_n}\big)$$

.

Вся програмная реализация была выполнена на языке Python 2.7.9, с использованием библиотек: nltk[12], gensim[13], pandas[14], sklearn[15], xgboost[16], а также библиотек, которые были указаны, в описаниях методов.

Для обучения моделей применялись следующие алгоритмы: XGBClassifier из библиотеки xgboost, RandomForestClassifier, LogisticRegression и svm.SVC из библиотеки sklearn. Рассмотрим параметры, которые настраивались для каждой модели: XGBClassifier – learning_rate, n_estimators; RandomForestClassifier – n_estimators; LogisticRegression – C, svm.SVC – C. Остальные параметры были настроены по умолчанию. Параметры подбирались по сеткам. В качестве итоговых параметров выбирались те, при которых качество модели является наилучшим, или перестает существенно улучшаться относительно критерия качества AUC. В качестве итоговой модели выбиралась та, которая дает наилучшее качество по AUC критерию.

[12] `http://www.nltk.org/`
[13] `https://radimrehurek.com/gensim/`
[14] `http://pandas.pydata.org/index.html`
[15] `http://scikit-learn.org/stable/`
[16] `http://xgboost.readthedocs.io/en/latest/`

Введем следующие обозначения для рассмотренных алгоритмов: алгоритм, основанный на парсинге текста – A_{par}, алгоритм, признаковое пространство которого основано на структуре предложений – A_{str}, алгоритм, признаковое пространство которого основано на контрасте эмоций и согласованности текста – A_{em}, алгоритм, признаковое пространство которого основано на использовании модели word2vec – A_{w2v}, алгоритм, признаковое пространство которого состоит только из n-грамм($n = 1, 2, 3$) – A_n, алгоритм, признаковое пространство которого основано на причинах появления сарказма – A_r, алгоритм, признаковое пространство которого состоит из всех признаковых пространств, рассмотренных, в предыдущих разделах – A_{all}. Рассмотрим результаты экспериментального исследования данных алгоритмов, которые приведены в таблице 1.

	A_{par}	A_{str}	A_r	A_{em}	A_{w2v}	A_n	A_{all}
F-мера:	0.30646	0.744748	0.74732	0.74942	0.78316	0.73130	0.80157
Recall:	0.20276	0.80628	0.78199	0.75796	0.80579	0.72311	0.83668
Precision:	0.62948	0.69224	0.71628	0.74173	0.76246	0.74076	0.76973
Accuracy:	0.54185	0.72390	0.73569	0.74675	0.77712	0.73451	0.79303
AUC:		0.79726	0.81229	0.82432	0.85644	0.81365	0.87662

Таблица 1: Результаты работы алгоритмов распознавания сарказма в тексте

Как видно из таблицы 1, алгоритм A_{par} показывает очень низкие результаты. Особенно низким является значение полноты у данного алгоритма. Это связано с тем, что данный алгоритм полагается на лексику, которая была составлена по собранной коллекции данных, и не позволяет предсказывать сарказм в предложениях, которые состоят из слов, отличных от тех, которые содержатся в изученном наборе. Кроме того, данный алгоритм полагается только на определенный набор слов в предложениях и никак не учитывает пунктационные и эмоциональные характеристики. Низкую точность можно связать с тем, что оценка тональности фразы (sentiment_score) является несовершенной. Так например, словосочетание "not good" будет определено как положительное, в связи с тем что оценка для слова "good" будет равна 0.7, а оценка для "not" будет равняться 0.

Наилучший результат для признакового пространства, основанного на структуре предложений (A_{str}) показал RandomForestClassifier(n_estimators = 1000) из библиотеки sklearn. Как видно из таблицы 1, результат работы данного алгоритма значительно лучше, чем A_{par}, так как он позволяет выявить признаки, которые определяют структуру предложения, которое содержит сарказм, однако, как видно из таблицы точность (Precision) является достаточно низкой, в то время как полнота высокая, что позволяет думать что алгоритм переобучается и настраивается на признаки, которые не являются истинными показателями сарказма.

Наилучший результат для признакового пространства, основанного на причинах появления сарказма в тексте (A_r) показал XGBClassifier(learning_rate = 0.06, n_estimators = 380) из библиотеки xgboost. Как видно из таблицы 1, ре-

зультаты работы данного алгоритма по AUC и F-мере лучше, чем результаты алгоритма (A_{str}), достигается это за счет увеличения точности алгоритма, то есть алгоритм точнее распознает случаи появления сарказма в тексте, однако упускает некоторые предложения с чем связана более низкая полнота.

Наилучший результат для признакового пространства, основанного на контрасте эмоций и согласованности текста (A_{em}) показал LogisticRegression($C = 0.6$) из библиотеки sklearn. Как видно из таблицы 1, результат работы данного алгоритма по AUC и F-мере и точности (accuracy) лучше, чем результаты алгоритма (A_{str}) и результаты алгоритма (A_r), также стоит отметить, что данный алгоритм позволяет получить одинаково неплохую полноту и точность(precision). Данный алгоритм имеет существенное отличие от предыдущих 2 в том, что использует n-граммы в качестве подмножества признаков, видимо этот факт позволяет ему распознавать сарказм с одинаково-хорошей точностью и полнотой.

Наилучший результат для признакового пространства, основанного только на n-граммах($n = 1, 2, 3$)(A_n) показал классификатор LogisticRegression($C = 1.3$) из библиотеки sklearn. Следует заметить, что результаты работы являются очень неплохими, однако стоит отметить что n-граммы никак не выражают внутреннюю структуру предложений, и могут давать очень плохой результат при несовпадении лексики на тестовой и обучающей выборке.

Наилучший результат для признакового пространства,, основанного на контрасте эмоций и согласованности текста (A_{w2v}) показал svm.SVC($C = 2.1$) из библиотеки sklearn. Как видно из таблицы результаты работы данного алгоритма по показателям являются лучшими, чем результы работы предыдущих алгоритмов, поэтому можно сделать вывод, что word2vec позволяет лучшим образом выявить признаки, которые определяют наличие сарказма в тексте.

Для признакового пространства, которое было получено в результате объединения всех описанных в данной работе признаковых пространств, при этом не содержащее повторяющихся признаков, классификатором который показал наилучшее качество является XGBClassifier(learning_rate $= 0.05$, n_estimators $=$ 600). Стоит отметить что данный подход позволил улучшить наилучшее качество, которое было достигнуто на едином признаковом пространстве.

Для того, чтобы улучшить качество классификации, исследовались композиции алгоритмов, которые были получены одним из следующих образов: композиция алгоритмов, основанная на голосовании по большинству: присваивает значение 1 результату, если большинство алгоритмов выдало 1, в противном случае присваивается 0 (обозначение : M); голосование с весами: присваивает 1 значению результату в том случае, если сумма весов за класс 1 больше суммы весов за класс 2, в противном случае результат равен 0 (обозначение: W). Результаты композиций алгоритмов представлены в таблицах 2, 3, 4, 5 (в случае использования в качестве композиции схемы голосования рядом с обозначением алгоритма в квадратных скобках записан его его вес). В ходе исследования перебирались всевозможные допустимые веса и различные комбинации алгоритмов. Так как наилучшими алгоритмами являлись A_{w2v} и A_{all}, то рассматривались только те

композиции алгоритмов, которые их содержат. В таблицах указаны только те результаты, которые позволили улучшить качество классификации по одному из критериев качества относительно алгоритма с максимальным качеством среди алгоритмов в композиции.

	M(A_{w2v}, A_{str}, A_{em})	M(A_{w2v}, A_{str}, A_n)	W(A_{w2v}[2], A_{str}[1], A_n[1])
F-мера:	0.78902	0.78657	0.77666
Recall:	0.82825	0.81853	0.77373
Precision:	0.75375	0.75765	0.78024
Accuracy:	0.77873	0.77814	0.77785

Таблица 2: Качество классификации сарказма при использовании композиции алгоритмов.

	W(A_{w2v}[2], A_{str}[1], A_r[1], A_{em}[1])	W(A_{w2v}[3], A_r[1], A_n[1], A_{em}[1], A_{str}[1])
F-мера:	0.79005	0.78847
Recall:	0.82632	0.81920
Precision:	0.75737	0.76073
Accuracy:	0.78050	0.78036

Таблица 3: Качество классификации сарказма при использовании композиции алгоритмов.

	W(A_{w2v}[2], A_{str}[1], A_n[1], A_r[1])	W(A_{all}[3], A_{w2v}[1], A_n[1], A_r[1])
F-мера:	0.78823	0.80121
Recall:	0.81895	0.82457
Precision:	0.76059	0.77964
Accuracy:	0.78006	0.79554

Таблица 4: Качество классификации сарказма при использовании композиции алгоритмов.

	W(A_{all}[4], A_{w2v}[1], A_n[1], A_r[1], A_{em}[1])	W(A_{all}[3], A_{w2v}[1], A_r[1], A_{em}[1])
F-мера:	0.80122	0.80070
Recall:	0.82752	0.82544
Precision:	0.77699	0.77786
Accuracy:	0.794808	0.79466

Таблица 5: Качество классификации сарказма при использовании композиции алгоритмов.

Как видно из таблиц, наилучшее качество классификации по f-мере (0.80157) можно достичь используя алгоритм на объединенном признаковом пространстве, никакая композиция алгоритмов не улучшила данный результат. Также следует отметить, что алгоритм на объединенном признаковом пространстве работает лучше, чем композиции отдельных алгоритомов на каждом из признаковых пространств. Наилучшее качество классификации по мере accuracy (0.79554) можно достичь используя композицию алгоритмов: W(A_{all}[3], A_{w2v}[1], A_n[1], A_r[1]).

7 Заключение

В ходе данной работы были получены следующие результаты:

- Реализовано 5 различных существующих методов распознавания сарказма в тексте.
- Предложено новое признаковое пространство, основанное на технологии word2vec.
- Собран набор данных, который может быть использован для исследования методов распознавания сарказма в тексте.
- Выделен метод, который показывает самое низкое качество. Это метод основанный на лингвистической структуре сарказма.
- Из существующих методов распознавания сарказма наилучшее качество показывает метод, основанный на оценке эмоциональной окраски и согласованности текста
- Новый метод, основанный на технологии word2vec показал наилучший результат среди всех реализованных алгоритмов, включающих в себя единое признаковое пространство.
- Метод, основанный на объединении всех признаковых пространств позволил существенно улучшить качество распознавания сарказма и получить наилучшее качество по F-мере: 0.80157 и AUC: 0.87662.
- При использовании композиций различных алгоритмов удалось получить наилучшее качество точности(accuracy): 0.79554.

Стоит заметить, что качество методов может быть занижено из-за того, что распознавание сарказма является сложной задачей даже для человека, в том случае, если для распознавания сарказма требуется знание личных предпочтений пользователя, или специализированной информации, известной только узкому кругу лиц.

Список литературы

[1] Bharti S. K., Babu K. S., Jena S. K. Parsing-based Sarcasm Sentiment Recognition in Twitter Data //Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015. – ACM, 2015. – С. 1373-1380.

[2] Barbieri F., Saggion H., Ronzano F. Modelling sarcasm in twitter, a novel approach //ACL 2014. – 2014. – С. 50.

[3] Miller G. A. WordNet: a lexical database for English //Communications of the ACM. – 1995. – Т. 38. – №. 11. – С. 39-41.

[4] Esuli A., Sebastiani F. Sentiwordnet: A publicly available lexical resource for opinion mining //Proceedings of LREC. – 2006. – Т. 6. – С. 417-422.

[5] Rajadesingan A., Zafarani R., Liu H. Sarcasm detection on twitter: A behavioral modeling approach //Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. – ACM, 2015. – С. 97-106.

[6] Tungthamthiti P., Kiyoaki S., Mohd M. Recognition of sarcasm in tweets based on concept level sentiment analysis and supervised learning approaches //Proceedings of Pacific Asia Conference on Language, Information and Computing, Phuket, Thailand. – 2014.

[7] Mikolov T. et al. Efficient estimation of word representations in vector space //arXiv preprint arXiv:1301.3781. – 2013.

[8] Godin F. et al. Multimedia Lab@ ACL W-NUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations //ACL 2015 Workshop on Noisy User-generated Text. – Association for Computational Linguistics, 2015. – С. 146-153.

[9] Thelwall M., Buckley K., Paltoglou G. Sentiment strength detection for the social web //Journal of the American Society for Information Science and Technology. – 2012. – Т. 63. – №. 1. – С. 163-173.