Санкт-Петербургский государственный университет

Кафедра компьютерного моделирования и многопроцессорных систем

Цыпушкин Арсений Витальевич

Применение методов машинного обучения для полуавтоматической обработки новостей

Выпускная квалификационная работа бакалавра

Направление 010300
Фундаментальная информатика и информационные технологии

Научный руководитель:
к. ф.-м. н., доцент
Корхов В. В.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Department of Computer Modelling and Multiprocessor Systems

Tsypushkin Arseniy

Bachelor's Thesis

Application of machine learning methods for semi-automatic news processing

Field of study 010300
Fundamental Informatics and Information Technology

Scientific supervisor:
Ph.D., Associate Professor
Korkhov V.V.

Saint-Petersburg
2017

Оглавление

Введение

В мире каждый день происходят преступления - криминал, терроризм, мошенничества и прочее. Для их предотвращения необходимо анализировать все события, которые происходят вокруг, иметь базу знаний, в которой события будут структурированы.

Встает необходимость наличия инструмента для анализа информации, которая поступает в большом количестве посредством новостей, на предмет наличия в ней преступного контекста с возможностью полуавтоматической обработки и сохранения результатов.

Проект W6 assess был образован для создания такого инструмента для некоммерческой компании Insecurity Insight, которая занимается сбором и анализом информации о преступлениях.

Компания Insecurity Insight анализирует:

- природу и паттерны человеческих отношений при вооруженных конфликтах
- преступления, связанные с сексуальным насилием, криминалом и восстаниями людей
- оружие, которое используют преступники
- преступления, нацеленные на определенные категории профессий - журналистов, социальных работников и прочих.

Результатом их работы являются:

- планирование и методики обеспечения безопасности
- получения связей между человеческими мероприятиями (выборы, медицинское обслуживание) и присутствием там насильственных событий

Проект W6 assess нацелен на получение 6 главных частей преступления из текста - что, где, когда, кто сделал, над кем и с помощью какого оружия. Это облегчит разбор больших статей и сразу выделит важные атрибуты преступления.

В результате проект W6 assess представляет из себя веб приложение, которое позволяет загружать новостные статьи, получать 6 частей преступления, сохранять их в необходимом формате.

Постановка задачи

Основная задача исследования - реализовать алгоритм поиска 6 частей преступления - что, где, когда, кто сделал, над кем и с помощью какого оружия. Цель данной работы состоит в исследовании алгоритмов машинного обучения и их готовых реализаций для применения к задаче поиска 6 частей преступления. Реализация исследования предполагает решение следующих задач:

- Изучение существующих библиотек для анализа текста
  - Методы анализа предложений
  - Кластеризация слов по частям речи
  - Построения дерева разбора предложения
  - Кластеризация слов по внешним признакам
  - Стемминг
- Изучение методов векторизации слов
  - Алгоритм word2vec
  - Алгоритм glove
- Поиск и разметка данных для анализа алгоритмов
- Реализация алгоритмов поиск места, времени, преступника и жертвы.
- Бинарная кластеризация глаголов по наличию признака - насильственный глагол
- Бинарная классификация существительных по наличию признака - оружие

Обзор литературы

В статье [20] описано применение метода CRF для задачи поиска частей речи в русскоязычных текстах. Исследование в статье показало, что CRF хорошо подходит для различных задачи распознавания сущностей в текстах.

В работе [8] подробно расписаны все виды зависимостей, используемых алгоритмом Dependency tree parsing, описанном в главе 1.1.3. На каждый тип зависимости приведены примеры использования в английском языке, описаны разные стили представления зависимостей. Присутствуют примеры использования библиотеки Stanford CoreNLP для получения дерева зависимостей.

Авторы статьи [5] описывают построение модели векторного пространства для задачи векторизации слов. Авторы соединили преимущества двух подходов: локального контекста окна (подробное описание в главе 2.1) и метод глобальной факторизации матриц. По результатам сравнения существующих моделей, полученная авторами модель Glove показывает неплохие результаты как в задаче поиска похожих слов, так и в задаче Named Entity Recognition.

1. Парсеры текста. Обзор и сравнение готовых решений.

1.1. Основные части библиотек для обработки текста

В данном пункте будут рассмотрены основные составляющие библиотек для обработки текста, примененные для решения задачи.

1.1.1. Named Entity Recognition

Определение. Named Entity Recognition - метод разметки и классификации именованных заранее частей - классов в тексте. Количество классов заранее известно. Классами могут являться имена, организации, места, время, параметры количества, денежные единицы, и другие.

Большинство существующих NER классификаторов, таких как Stanford Named Entity Recognizer, основано на алгоритме CRF (Conditional random field). CRF - разновидность метода скрытых Марковских моделей. Алгоритм рассматривает условное распределение $(y|x)$ последовательности классов $y \in Y$, где $x \in X$ - вектор из рассматриваемых элементов . Из рассматриваемых и выходных результатов формируется набор потенциальных функций, включающих в себя произвольное количество элементов. В графовом представлении каждая потенциальная функция, определенная на графе $G = (V, E)$ (Марковском случайном поле), каждой клике из связанных элементов в графе ставит в соответствие неотрицательное вещественное число. Тогда условное случайное поле - распределение вида:

$$p(y|x) = \frac{1}{Z(X)} * \prod_i exp(\sum_i \lambda_i f_i(y_m, y_{m-1}, x_m)),$$

где f - функция-признак, λ - множитель Лагранжа, Z - коэффициент нормализации. Результат - это сумма по всем $y \in Y$. Вычисление результата происходит как решение оптимизационной задачи с огра-

ничениями, такими как $\sum_{y \in Y} p(y|x) = 1$. Для вычисления используются алгоритмы “forward-backward” и Витерби. Более подробное описание присутствует в статье [20].

В статье [20] представлено сравнение CRF с другими методами для использования в распознавании лингвистических классов по мере F1, описание которой представлено в 3.3.2.

В данный момент, самые лучшие системы NER классификации, такие как MUC-7, выдают точность 93.3

1.1.2. POS tagger

Определение. POS tagger - (part of speech tagger) алгоритм разметки слов в текстах на предмет определения частей речи. Существуют 2 подхода для построения POS таггера: подход на основе правил (rule-based) и стохастический подход.

Подход на основе правил.

В первом подходе используется набор правил, который составляется вручную. Применяется, если для одного слова есть больше одного тэга. Для разметки используются различные признаки, такие как тэги ближайших слов, лингвистические особенности слова.

Стохастические модели построения таггера.

В стохастическом подходе для построения таггеров используются модели Маркова. Для обучения таких таггеров используются скрытые модели, основной особенностью которых является скрытость промежуточных состояний. Основными алгоритмами, которые применяются в данном подходе, являются алгоритмы Витерби, алгоритм поиска модели с наибольшей энтропией, алгоритм Баума-Вэлша, transformation-based алгоритм (применяемый в таггере Бриля).

1.1.3. Dependency tree parser

Основное понятие парсинга зависимостей - treebank.

Определение. Treebank - это текстовый корпус, размеченный по семантике аннотаций и семантическим структурам. В виде аннотаций могут быть использованы части речи (POS taging), в качестве семантики выступают языковые семантические структуры разных языков.

Для создания данного банка деревьев могут быть использованы следующие подходы:

- Ручная разметка. Этот подход предполагает использование разметку корпуса вручную. Определение необходимых аннотаций и связей между словами в предложении.
- Разметка с применением правил. Данный подход использует заранее подготовленный набор правил, сформированных в виде - $A \to B\ C$, где A, B, C могут являться какими-либо аннотациями (частями речи, семантическими структурами)
- Использование рекурсивных нейронных сетей для построения грамматических структур.

Каждый treebank имеет свои типы связей и аннотаций. Связи используются для построения семантической структуры текста, аннотации используются для пометки слов в тексте.

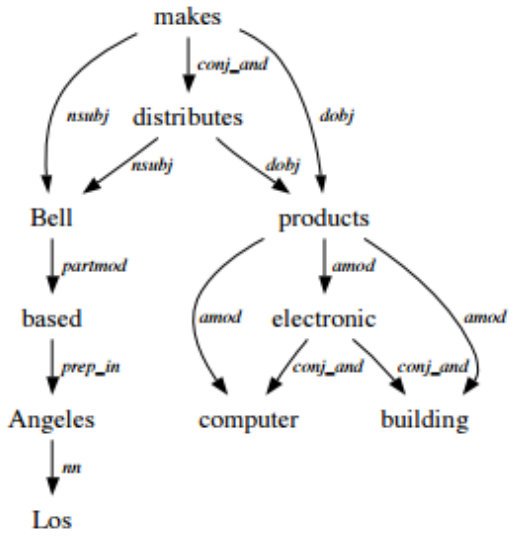

Рис. 1: Пример разбора предложения

На основе существующих банков деревьев существуют парсеры, которые размечают тексты на основе правил, описанных в банке деревьев. Описание парсеров представлено в пункте 1.2.

1.1.4. Stemming

Определение. Stemming - процесс нахождения основы слова.

Данный процесс необходим для проверки слов на эквивалентность по значению. Рассмотрим алгоритмы Портера и Пэйса/Хаска.

Алгоритм Портера

Данный алгоритм стемминга является контекстно чувствительным относительно удаления суффиксов. Алгоритм делится на 5-6 шагов. Слово в алгоритме представлено в виде $[C](VC)m[V]$, где $[C]$ - приставка, $[V]$ - суффикс и окончание, m - мера повторения основы (VC), которая всегда больше равна 0. Далее к слову применяются итеративно правила для удаления $[C]$, $[V]$ и превращения m в 0.

Алгоритм Пэйса/Хаска

Данный алгоритм нацелен на итеративное удаление окончания слова. В алгоритме применяется таблица правил для замещения окончаний и суффиксов. Алгоритм опирается на последнюю букву в слове, что делает эффективным поиск правил в общей таблице. Удаление идет до тех пор, пока не существует правил в таблице, подходящих для данного слова.

1.2. Обзор существующих библиотек для анализа текстов

SYNC3 - платформа для анализа новостей, предназначенная для журналистов. Платформа загружает статьи из разных источников, проводит кластеризацию по событиям. Собирает информацию как с официальных источников, так и с блогов обычных людей.

Программа помогает составить независимые новости, опираясь на мнения не только источников, но и отзывы на отзывы людей. Платформа имеет сервис-ориентированную архитектуру, restfull сервера, которые имеют разные уровни (tier). Платформа очень гибкая, расширяемая и имеет апи для интеграции в любые другие сервисы.

Xerox Incremental Parser - лингвистический парсер текста. Делает разбор текста по частям речи, умеет выделять важные части текста, строит диаграмму зависимостей между словами (дерево разбора предложения). Имеет rest-api для интеграции в любые другие сервисы. Умеет работать с 3-мя языками - английским, французским, немецким. Не open-source решение.

Apache OpenNLP - лингвистический парсер текста. Состоит из NER таггера, определителя предложений, классификатора документов, POS таггера, лемматизатора, и других частей. Для каждого из методов для анализа существует API для возможности собственного обучения модели алгоритма. Написан на языке Java. Данная библиотека распро-

страняется под лицензией Apache Licence, v2.0, что позволяет любому человеку участвовать в разработке продукта.

Stanford CoreNLP - фреймворк для обработки текстов формального языка. Одно из самых популярных open source решений для анализирования текстов.

Сама библиотека состоит из разных других проектов Stanford NLP Group, таких как: POS таггера, NER таггера, Statistical Parser - статистического грамматического парсера, Stanford Open Information Extraction - средство для выделения бинарных связей частей предложения и других. Распространяется под лицензией GNU.

Данная библиотека была выбрана для использования в разработке алгоритмов, описанных в главе 3, из-за следующих причин:

- API для многих современных языков программирования
- Очень подробная документация с примерами
- Полный необходимый семантический анализ - разбор предложения по частям речи, нормализация слов, получение дерева разбора предложения.
- Много поддерживаемых языков - en, de, fr, ch.
- Open Source решение

NLTK - лингвистический парсер текста, предназначенный для анализа и обрабки текстов на языке Python. Имеет такие же возможности, что и Apache OpenNLP и Stanford CoreNLP. Распространяется под лицензией Apache License v2.0. Библиотека была выбрана для создания классификатора, описанного в главе 3, по причине наличия готовых библиотек для работы с обученными моделями Word2Vec и Glove, описанных в главе 2.

2. Методы машинного обучения для векторизации слов

2.1. Word2Vec

Определение. Word2Vec - это набор моделей для векторизации слов. На вход данной модели подается корпус текстов, на выходе получается набор векторов заданной размерности.

Данная модель применяется для поиска слов, близких по контексту, то есть семантически. Близость в данной модели определяется считается по косинусному расстоянию.

Задача построения модели Word2Vec - максимизация расстояния между словами, которые не встречаются рядом друг с другом и минимизация расстояния между словами, которые встречаются друг с другом.

Введем основные понятия, которые участвуют в построении модели.

Определение. Окно размерности n - набор из n подряд идущих слов.

Определение. Опорное слово окна - центральное слово.

Определение. Субпредложение - базовый элемент корпуса. Обычно субпредложением выступает предложение, но может также быть абзац или целая статья.

Определение. Субсэмплирование - удаление самых частых слов для увеличения качества модели.

Архитектуры построения нейронных сетей.

В Word2Vec применяются две основные архитектуры - CBOW [2] и Skip-gram [18]. Данные архитектуры применяются для обучения нейронных сетей. Отличаются данные архитектуры следующим - CBOW делает предсказание при данном контексте о следующем слове, а Skip-Gram на основе данного слова предсказывает контекст.

Рассмотрим подробнее каждый из них.

Определение. CBOW (Continuous Bag of Words) - модель “мешка” слов с учетом размера окна.

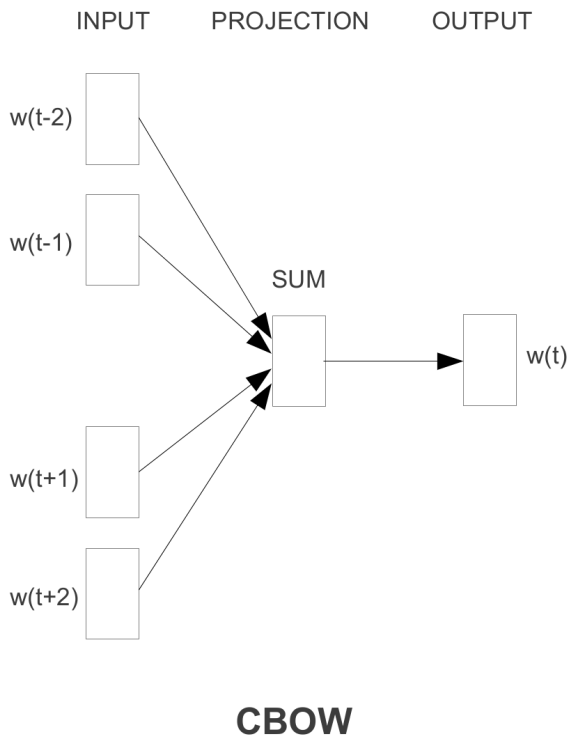

Рис. 2: Принцип построения модели CBOW

Определение. Skip-gram (k-skip-n-gram) - для создания последовательности длиной n берутся слова, находящиеся на расстоянии, не превосходящем k друг от друга.

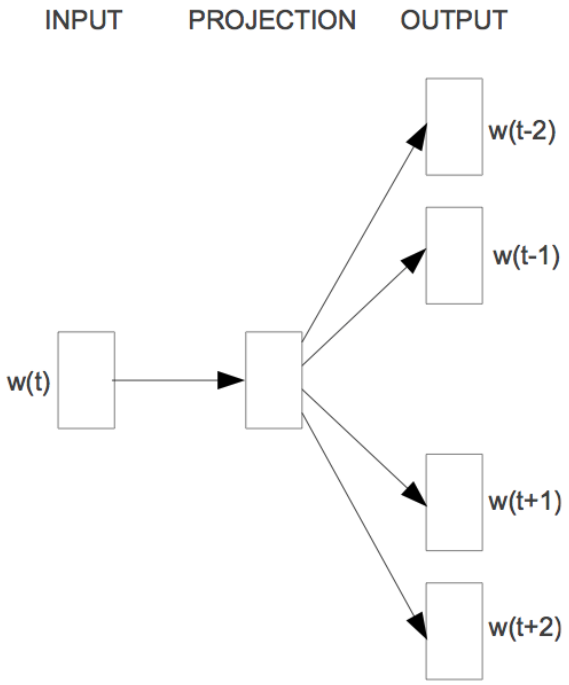

Рис. 3: Принцип построения модели Skip-gram

Для того, чтобы сократить перебор всех контекстов, существует алгоритм негативного сэмплирования, в котором все возможные контексты опорного слова заменяются на случайно выбранные. Это позволяет уменьшить вычислительные операции для ложных контекстов данного слова в процессе обучения модели.

Основные этапы построения модели.

1. На вход модели подается корпус документов. Рассчитывается встречаемость каждого слова в корпусе.

2. Слова сортируются по частоте и происходит удаление редких слов.

3. Строится словарь слов. Для этого применяется код Хаффмана для построения префиксного дерева.

4. Для каждого субпредложения в корпусе производится субсэмплирование.

5. По каждому субпредложению происходит создание n-грамм по заранее заданному размеру окна для дальнейшего применения моделей CBOW или Skip-gram.

6. Создается нейросеть прямого распространения по одной из моделей Skip-gram или CBOW. В роли функции активации используется иерархический софтмакс и/или негативное сэмплирование.

2.2. Glove

Определение. Glove [3] - алгоритм машинного обучения без учителя для векторизации слов.

Основные этапы построения модели.

Описание алгоритма.

1. Создается матрица A_{ij} встречаемости каждого слова с каждым. Как часто слово i встречается в контексте слова j. Для каждого слова в корпусе используются близкие слова, находящиеся до данного слова на размер окна и после на размер окна. Может добавляться вес слова как величина, обратная расстоянию между словами в окне.

2. Определяется мягкое ограничение для каждой пары слов, где ω_i - главное слово, ω_j - слово в контексте, b_i, b_j - дополнительные отклонения.
$$\omega_i^T \omega_j + b_i + b_j = log(A_{ij})$$

3. Создается функция стоимости
$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(A_{ij})(\omega_i^T \omega_j + b_i + b_j - log(A_{ij}))^2,$$
где f - весовая функция, необходимая для предотвращения обучения только для самых близких пар слов. Создателями алгоритма была выбрана функция:
$$f(x) = \begin{cases} (\frac{x}{x_{\max}})^\alpha & \text{if } x < x_{\max} \\ 1 & otherwise \end{cases}$$

Данный алгоритм обучения выдает хорошую точность определения близости слов на метриках MUC 7, ACE. Подробное сравнение алгоритмов приведено в статье [5].

3. Реализация задачи

3.1. Подготовка данных

Для того, чтобы искать 6 частей преступления в новостях, встала необходимость получить датасет новостных статей, основной темой которых будет криминал и преступления. Для этого был использован открытый новостной ресурс “The guardian”, в котором присутствует рубрика “crime”. Из данного ресурса было получено 25 тысяч статей про преступления. Статьи были получены с помощью парсинга HTML страниц. Было получена основная текстовая информация в статье, без заголовка, и сохранена в формате txt.

3.2. Разметка полученных данных

Для проверки и сравнения работы дальнейших алгоритмов встала задача разметки данного датасета на предмет наличия в статьях 6 частей преступления. Разметка была произведена вручную мной, как экспертом. В результате было получено 250 размеченных статей по критериям 6 частей преступления.

3.3. Метрики для оценки точности алгоритмов

Для оценивания работы алгоритмов вставала задача выбора оптимальных метрик для следующих задач:

- Сравнение словосочетаний для задач поиска места, времени, преступника и жертвы
- Двух-классовой классификации глаголов и существительных на классы насильственных глаголов и оружия.

3.3.1. Метрика для сравнения словосочетаний

Сравнение словосочетаний опирается на сравнение слов. Для сравнения слов существуют следующие метрики: расстояние Левенштейна, расстояние Хэмминга, сходство Демерау - Левенштейна, расстояние Джаро - Винклера.

Расстояние Левенштейна.

Расстояние Левенштейна - минимальное количество операций "замены", "удаления" и "вставки" для превращения одной строки в другую.

Данная метрика применяется в задачах сравнения файлов в операционных системах Unix (команда "diff"), исправления ошибок в словах в различных поисковых системах, базах данных, для сравнения геномов, белков и хромосом в биоинформатике.

Описание алгоритма:

Пусть у нас имеются 2 строки A и B над некоторым алфавитом длины n и m соответственно. Тогда расстоянием между ними $d(A, B) = D(A, B)$, где

$$D(i,j) = \begin{cases} 0, & \text{if } i = 0,\ j = 0 \\ i, & \text{if } i > 0,\ j = 0 \\ j, & \text{if } i = 0,\ j > 0 \\ min(& \\ \quad D(i-1,j), & \\ \quad D(i,j-1), & \text{if } i > 0,\ j > 0 \\ \quad D(i-1,j-1) + m(S_1[i], S_2[j]) & \\ ) & \end{cases}$$

где $m(a, b) = 0$, если $a = b$, иначе $= 1$.

В данном случае, выбор j символизирует вставку в первую строку, i - удаление из первой строки и замену символа в последнем случае.

Основными недостатками применения чистого метода являются:

- Большое расстояние между словами, полученными перестановкой своих частей.
- Между небольшими словами расстояния будут небольшие, между большими похожими словами расстояние будет большое.

В последующих алгоритмах будет использована усовершенствованная метрика Левенштейна, которая будет принимать словосочетания, делить их на слова, сравнивать наборы слов стандартным алгоритмом и возвращать близость словосочетаний в пределах от 0 до 100. Алгоритм был взят из библиотеки fuzzywuzzy [6].

Расстояние Джаро — Винклера.

Данное расстояние состоит из 2х частей - расстояния Джаро и дополнительного коэффициента масштабирования.

Расстояние Джаро D_j между двумя строками S_1 и S_2:

$$D_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}) & otherwise \end{cases}$$

где $|S|$ - длина строки, m - число совпадающих символов, t - половина числа транспозиций.

Расстояние Джаро-Винклера DW_j между двумя строками S_1 и S_2:

$$DW = D_j + (lp(1 - D_j)),$$

где D_j - расстояние Джаро, l - длина общего префикса от начала строки (максимум до 4х символов), p - коэфициент масштабирования, который отвечает за коррекцию оценки в сторону повышения для общих префиксов. Коэфициент масштабирования не должен превышать 0.25, обычно равен 0.1.

Для сравнения словосочетаний была выбрана метрики Левенштейна и Джаро — Винклера из-за того, что расстояние Хэмминга применяется

для строк одинаковой длины и расстояние Демерау-Левенштейна отличается от расстояния Левенштейна тем, что учитывает дополнительно транспозицию символов, что не имеет смысла в данном случае.

3.3.2. Метрики для определения качества бинарных классификаторов

Для определения качества бинарной классификации слов были использованы следующие характеристики:

- TPR
- FPR
- F1

Качественные характеристики классификации берут свое начало в таблице контингентности.

Категория		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

TP - true positive - истинно-положительные решение

TN - true negative - истинно-отрицательное решение

FP - false positive- ложно-положительное решение

FN - false negative - ложно-отрицательное решение

Эти переменные - количества результатов после классификации, которые попали в тот или иной класс решений.

Введем понятия точности(precision) и полноты(recall).

Определение. Точность классификатора - это доля тех слов, которые действительно принадлежит данному классу относительно всех слов, которые классификатор отнес к данному классу.

$$Precision = \frac{TP}{TP + FP}$$

Определение. Полнота классификатора - это доля найденных классификатором слов, которые принадлежат данному классу, относительно всех слов этого класса в тестовой выборке. Также полноту называют TPR (true positive rate).

$$Recall = \frac{TP}{TP + FN}$$

Определение. FPR (false positive rate) - число объектов из общего числа объектов, не попавших в наш класс, которые были отнесены в наш класс.

$$FPR = \frac{FP}{FP + TN}$$

Определение. F мера - гармоническое среднее между точностью и плотностью.

$$F = (\beta^2 + 1)\frac{Precision * Recall}{\beta^2 Precision + Recall},$$

где $\beta \in (0, 1]$. При $\beta = 1$ F мера называется сбалансированной ($F1$).

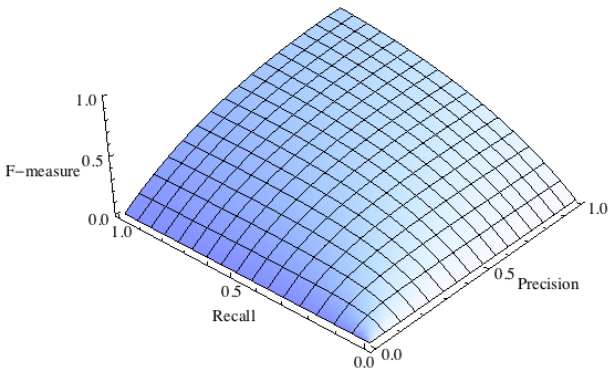

Рис. 4: Сбалансированная F мера

Также для оценки классификатора будем использовать ROC-кривую.

Определение. ROC-кривая - кривая, которая позволяет оценить бинарную классификацию на основе TPR и FPR.

Если классификатор позволяет оценить вероятность принадлежности объекта к нужному классу, то качественной оценкой кривой, построенной при разных значениях данной вероятности, можно считать AUC (area under curve) - площадь под графиком ROC-кривой.

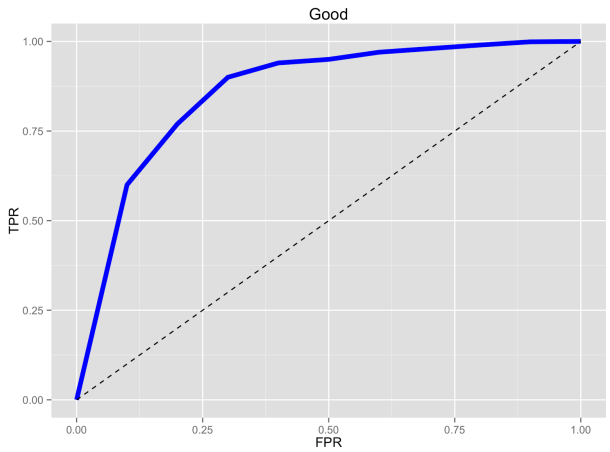

Рис. 5: ROC-кривая

3.4. Основной алгоритм поиска 6 частей преступления

Алгоритм основывается на парсинге дерева разбора предложения, которое выдает библиотека Stanford CoreNLP. На вход алгоритма подается дерево разбора предложения по типизированным зависимостям между словами в предложении, разбор слов в предложении по частям речи и инструмент для нормализации слов. Данный алгоритм будем в дальнейшем называть DTP (Dependency tree parsing).

Подробное описание дерева разбора предложения представлено в 1.1.3 , получение частей речи описано в 1.1.2.

Алгоритм основывается на наличии в предложении насильственного глагола. Поиск насильственного глагола происходит с помощью словаря насильственных глаголов, получение которого описано в части 3.7.

При наличии в предложении насильственного глагола, используя связь между подлежащим и сказуемым, подлежащим и дополнением, мы получаем преступника и жертву преступления. У нас может быть 2 случая - активный и пассивный залоги. Если глагол находится в активном залоге, то подлежащее является преступником, а дополнение - жертвой. Если глагол находится в пассивной форме, то наоборот - подлежащее - жертва, а дополнение - преступник. Чтобы получить контекст преступления, в качестве подлежащего и дополнения используется все поддерево, корнем которого является слово, связанное с глаголом. Таким образом получаются “словосочетания”, которые являются преступником и жертвой.

При отсутствии насильственного глагола в предложении, алгоритм сможет найти только оружие, место и время.

Для поиска времени и места в дереве разбора предложения ищутся связи обстоятельства времени и места. При наличии данных связей в дереве, алгоритм получает “словосочетания” из всего поддерева для получения контекста.

Для поиска оружия используется словарь оружия, получение и расширение которого описывается в пункте 3.7, нормализуются все суще-

ствительные и проверяются на присутствие в словаре.

Сложность алгоритма складывается из алгоритмов получения дерева разбора предложения, получения частей речи слов в предложении, алгоритма нормализации слов, умноженное на количество слов в предложении.

Узким местом алгоритма является зависимость от библиотеки, которая предоставляет функционал, описанный выше, сложность работы которой не всегда поддается корректному описанию.

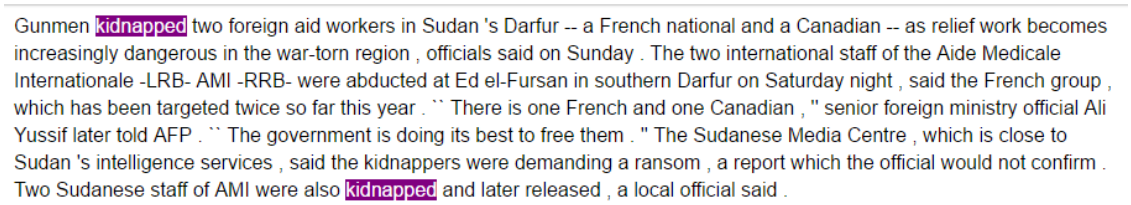

Рис. 6: Пример разбора текста, часть 1

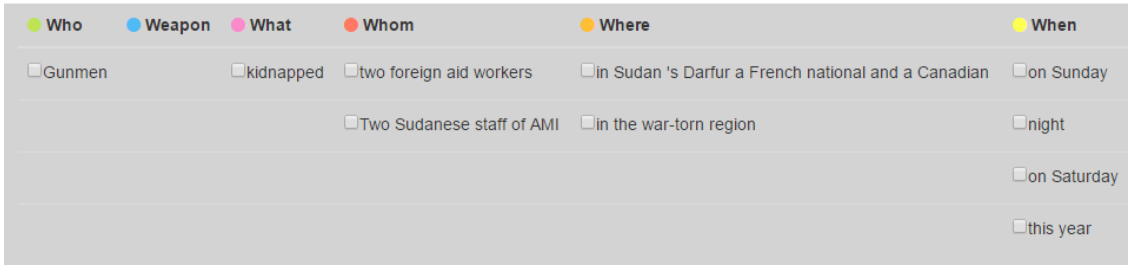

Рис. 7: Пример разбора текста, часть 2

3.5. Сравнение методов Named Entity Recognition и Dependency tree parsing для нахождения места и времени

Для поиска места и времени из новостных статей было выбрано 2 алгоритма для сравнения - Named Entity Recognition и реализован алгоритм на основе Dependency tree parsing с помощью библиотеки Stanford CoreNLP.

Реализация первого была взята из библиотеки Stanford CoreNLP. На вход подавались статьи из датасета, получение которого описано в пункте 3.1. На выход приходили статьи, каждое слово которых было помечено набором тэгов из данного алгоритма, подробное описание которых описано в главе 1.1. Нас интересуют только тэги “location” и “date” + “time”. Слова, помеченные данными тэгами, сохраняем для применения к ним метрик, описанных в пункте 3.3.

Реализация второго метода получения места и времени взята из основного алгоритма поиска 6 частей преступления с помощью разбора дерева зависимостей. На вход приходили статьи из датасета, описанного в пункте 3.1. На выход мы получали словосочетания, помеченные в дереве зависимостей предложения как обстоятельства времени и места. Для их дальнейшего анализа с помощью метрик, словосочетания делились на слова, убирались стоп слова и лишние знаки препинания.

Для анализа результатов был использован размеченный датасет, расстояние Левенштейна и Джаро - Винклера для сравнения наборов слов.

Метрика/Алгоритм	NER loc	NER d/t	DTP loc	DTP d/t
Левенштейн	0.81	0.84	0.51	0.7
Джаро-Винклер	0.75	0.67	0.57	0.55

3.6. Анализ результатов Dependency tree parsing для поиска преступника и жертвы

Для поиска преступника и жертвы из новостных статей был реализован алгоритм на основе Dependency tree parsing с помощью библиотеки Stanford CoreNLP.

Реализация метода получения преступника и жертвы взята из основного алгоритма поиска 6 частей преступления с помощью разбора дерева зависимостей. На вход приходили статьи из датасета, описанного в пункте 3.1. На выход мы получали словосочетания, помеченные в дереве зависимостей предложения как подлежащее и дополнение. Для их дальнейшего анализа с помощью метрик, словосочетания делились на слова, убирались стоп слова и лишние знаки препинания.

Для анализа результатов был использован размеченный датасет, расстояние Левенштейна и Джаро - Винклера для сравнения наборов слов.

Метрика/Алгоритм	DTP loc	DTP d/t
Левенштейн	0.25	0.505
Джаро-Винклер	0.57	0.62

3.7. Сравнение обученных моделей w2v и glove для расширения словарей насильственного глагола и оружия

Описание обученных моделей Word2Vec и Glove.

В бинарных классификаторах были использованы готовые матрицы слов размерности 300 * n, где n - количество слов в обученной модели. Матрица для модели w2v получена из открытого репозитория компании Google, матрица для модели glove была получена из открытого репозитория Стэнфордского университета.

Описание классификатора.

Классификатор построен на понятии близости между словами по моделям glove и w2v. Основой для сравнения новых слов были взяты словари насильственных глаголов и оружия, составленные вручную. Каждое новое слово сравнивается со всеми в словаре, берется максимум по этим величинам, и если он больше заранее заданного уровня, то это слово подходит по нашему критерию классификации. Данный классификатор был создан для расширения словарей.

Классификатор реализован на языке python с использованием библиотек nltk (стемминг, токенайзер) и gensim (работа с обученными моделями w2v и glove).

Далее приведены результаты метрик по каждой из моделей, полученные на датасете, получение которого описано в части 3.1. Подсчет метрик производился вручную.

Статистика для уровня близости слов - 0.6.

Метрика/Модель	w2v verbs	w2v weap	glove verbs	glove weap
TPR	0.99	0.99	0.99	0.98
FPR	0.19	0.2	0.06	0.17
F1	0.99	0.99	0.99	0.98

Построение roc кривой происходило по точкам, которые получены при разных значениях близости между словами - от 0.3 до 0.8. Данный метод построения был выбран из-за того, что получение tpr и fpr происходило вручную. Количество глаголов, использованных для классификации - 2806, из них насильственных 103, изначально в словаре - 54. Количество существительных, использованных для классификации оружия - 8622, из них оружия - 113, изначально в словаре 76.

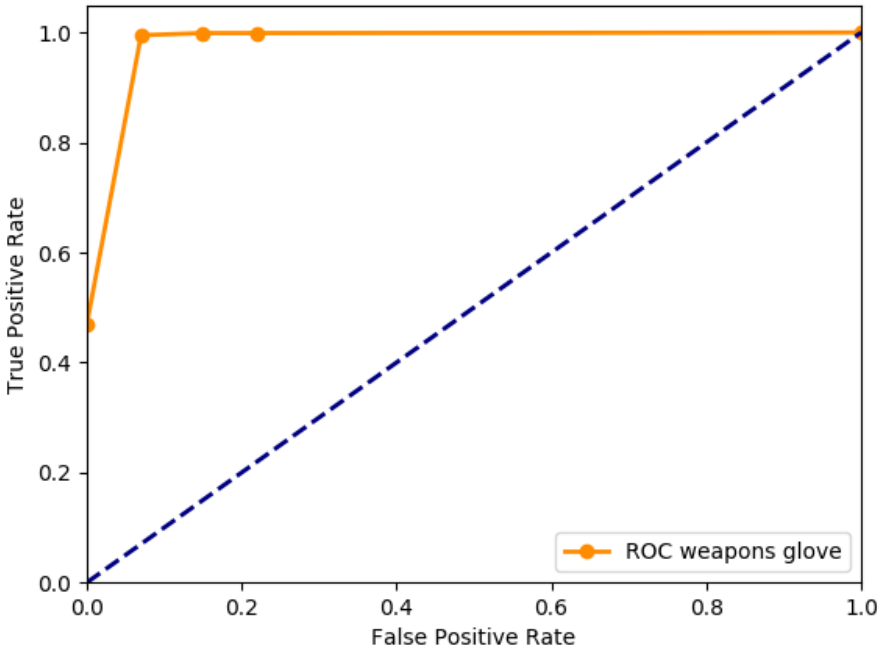

Рис. 8: Roc кривая для классификатора оружия модели glove

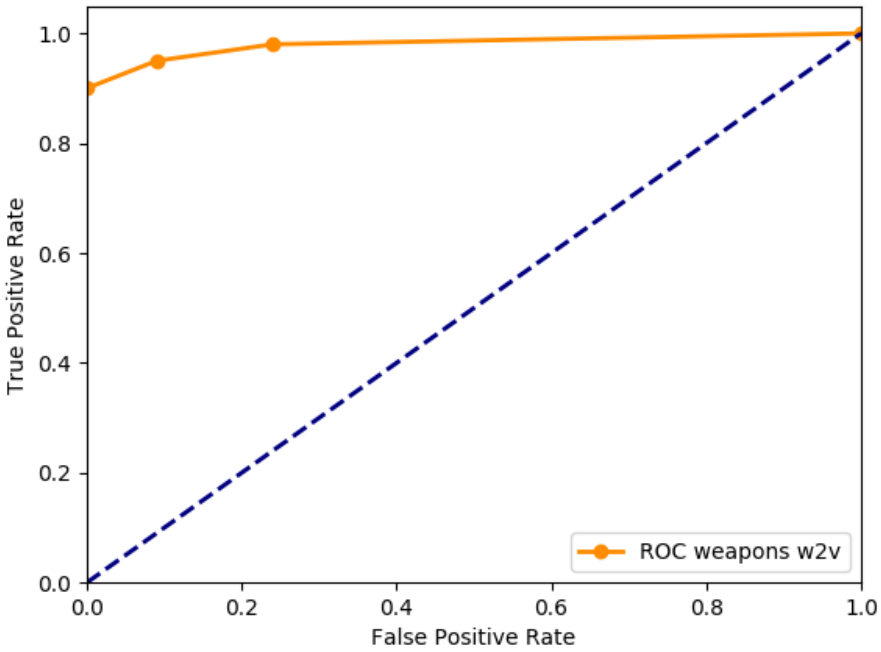

Рис. 9: Roc кривая для классификатора оружия модели w2v

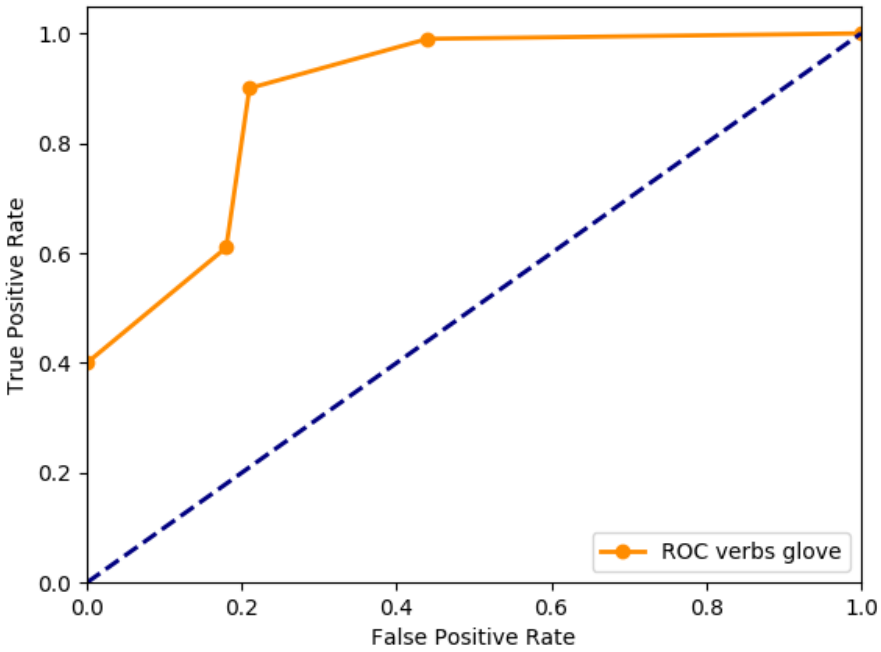

Рис. 10: Roc кривая классификатора насильственных глаголов модели glove

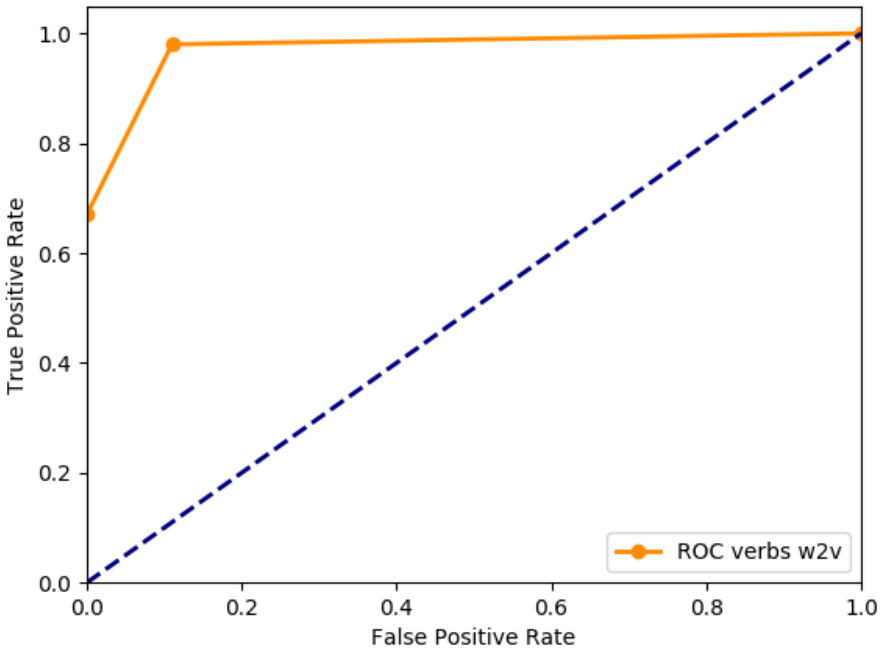

Рис. 11: Roc кривая классификатора насильственных глаголов модели w2v

Выводы

В данной работе был реализован алгоритм поиска 6 частей преступления с помощью функционала библиотеки Stanford CoreNLP на языке Java.

Для сравнений классификаторов и алгоритмов поиска различных частей преступления был создан датасет со статьями о преступлениях с открытого новостного ресурса Theguardian.

Для поиска места и времени были рассмотрены обученный алгоритм NER и алгоритм, основанный на обработке дерева зависимостей. Для сравнения алгоритмов были выбраны расстояния Левенштейна и Джаро - Винклера.

Для поиска преступника и жертвы был реализован алгоритм, основанный на разборе дерева зависимостей от насильственного глагола. Результаты подсчета метрик, полученные от данного алгоритма, показали, что его применение не уместно для данной задачи, так как описание преступника и жертвы чаще происходит не в подлежащем и дополнении.

Для создания классификаторов насильственных глаголов и оружия были использованы обученные модели w2v и glove. Результаты работы классификаторов представлены в виде roc кривых. Опираясь на анализ roc кривых можно сделать вывод, что классификаторы хорошо подходят для решения данных задач и выбор модели не сильно влияет на процесс классификации.

Заключение

В рамках данного исследования были получены следующие результаты:

- Обученный алгоритм NER лучше подходит для поиска места и времени в криминальных статьях, чем алгоритм, основанный на анализе дерева зависимостей.
- Алгоритм, основанный на анализе зависимостей не подходит для задачи поиска преступника и жертвы в криминальных статьях.
- Для расширения словарей насильственных глаголов и оружия можно использовать классификаторы, основанные как на обученной модели w2v, так и на модели glove.

Поставленные задачи были выполнены в полной мере.

В дальнейшем планируется создать алгоритм, который будет давать лучшие результаты для поиска преступника и жертвы.

Список литературы

[1] Apache OpenNLP. — Access mode: https://opennlp.apache.org/.

[2] Continuous Bag of Words (CBOW). — Access mode: https://iksinc.wordpress.com/tag/continuous-bag-of-words-cbow/.

[3] Global Vectors for Word Representation. — Access mode: https://nlp.stanford.edu/projects/glove/.

[4] Insecurity Insight Webpage. — Access mode: http://www.insecurityinsight.org/.

[5] J. Pennington R. Socher Chr. D. Manning. GloVe: Global Vectors for Word Representation. — Access mode: https://nlp.stanford.edu/pubs/glove.pdf.

[6] Java fuzzy string matching implemenation of the well known Python's fuzzywuzzy algorithm. — Access mode: https://github.com/xdrop/fuzzywuzzy.

[7] K. Toutanova Chr. D. Manning. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. — Access mode: https://nlp.stanford.edu/~manning/papers/emnlp2000.pdf.

[8] M-C. de Marneffe Chr. D. Manning. Stanford typed dependencies manual. — 2016. — Access mode: https://nlp.stanford.edu/software/dependencies_manual.pdf.

[9] M.H. Zweig G. Campbell. Receiver-Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine. — Access mode: http://clinchem.aaccjnls.org/content/clinchem/39/4/561.full.pdf.

[10] Natural Language Toolkit. — Access mode: http://www.nltk.org/.

[11] Paice/Husk Stemming Algorithm. — Access mode: https://web.archive.org/web/20140826000545/http://www.comp.lancs.ac.uk:80/computing/research/stemming/Links/paice.htm.

[12] Porter Stemming Algorithm. — Access mode: https://web.archive.org/web/20140826021336/http://www.comp.lancs.ac.uk:80/computing/research/stemming/Links/porter.htm.

[13] SYNC3 project. — Access mode: https://web.archive.org/web/20160630172138/http://www.sync3.eu/.

[14] Stanford CoreNLP – Core natural language software. — Access mode: https://stanfordnlp.github.io/CoreNLP/.

[15] Stanford Dependencies. — Access mode: https://nlp.stanford.edu/software/stanford-dependencies.shtml.

[16] Stanford Log-linear Part-Of-Speech Tagger. — 2016. — Access mode: https://nlp.stanford.edu/software/tagger.html.

[17] Stanford Named Entity Recognizer (NER). — 2016. — Access mode: https://nlp.stanford.edu/software/CRF-NER.shtml.

[18] Vector Representations of Words. — Access mode: https://www.tensorflow.org/tutorials/word2vec.

[19] Xerox Incremental Parser. — Access mode: https://open.xerox.com/Services/XIPParser.

[20] А. Ю. Антонова А. Н. Соловьев. Метод условных случайных полей в задачах обработки русскаязычных текстов. — Access mode: http://itas2013.iitp.ru/pdf/1569759547.pdf.