МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | **Автоматики и вычислительной техники** |
| Кафедра | **Автоматизированных систем управления** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оценка комиссии: |  | | | Рейтинг: |  |
| Подписи членов комиссии: | | | | | |
|  | |  |  | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | |  |  | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | | | | | |
| (дата) | | | | | |

**КУРСОВАЯ РАБОТА**

|  |  |
| --- | --- |
| по дисциплине | Методы и модели искусственного интеллекта |
| в задачах нефтегазового комплекса | |

|  |  |
| --- | --- |
| на тему | Системы обработки естественного языка в задачах |
| нефтегазового комплекса | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| «К ЗАЩИТЕ» |  | ВЫПОЛНИЛ: |  |
|  |  | Студент группы | **АСМ-22-04** |
|  |  |  | (номер группы) |
| профессор, д.э.н.; Алетдинова А.А. |  | Азаров Иван Сергеевич | |
| (должность, ученая степень; фамилия, и.о.) |  | (фамилия, имя, отчество) | |
|  |  |  | |
| (подпись) |  | (подпись) | |
|  |  |  | |
| (дата) |  | (дата) | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Москва, 20 | 22 |  |

МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | **Автоматики и вычислительной техники** |
| Кафедра | **Автоматизированных систем управления** |

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

|  |  |
| --- | --- |
| по дисциплине | Методы и модели искусственного интеллекта |
| в задачах нефтегазового комплекса | |

|  |  |
| --- | --- |
| на тему | Системы обработки естественного языка в задачах |
| нефтегазового комплекса | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| ДАНО студенту | **Азарову Ивану Сергеевичу** | группы | **АСМ-22-04** |
|  | (фамилия, имя, отчество в дательном падеже) |  | (номер группы) |

**Содержание работы:**

|  |  |
| --- | --- |
|  | Определение методов, необходимых для создания NLP-системы |
|  | Выбор библиотек для построения NLP конвейера |
|  | Практическая реализация одной из задач для НГК |
|  | особенности применения NLP в задачах НГК |
|  | Анализ полученных результатов и выводы |

**Исходные данные для выполнения работы:**

|  |  |
| --- | --- |
|  | Отчет об отключении блокировок противоаварийной защиты, с одного из предприятий АО «Роснефть» (как датасет) |
|  |  |
|  |  |

**Рекомендуемая литература:**

|  |  |
| --- | --- |
|  | Natural Language Processing in Action Understanding, analyzing, and generating text with Python» HOBSON LANE, COLE HOWARD, HANNES MAX HAPKE |
|  | Natural Language Processing with Python» Steven Bird, Ewan Klein, and Edward Loper |

**Графическая часть:**

|  |  |
| --- | --- |
|  | Приложение А |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель: | д.э.н. |  | профессор | | | |  |  | | |  | Алетдинова А.А. | |
|  | (уч.степень) |  | (должность) | | | |  | (подпись) | | |  | (фамилия, имя, отчество) | |
| Задание принял к исполнению: | | | | студент |  |  | | |  | Азаров И.С | | |
|  | | | | |  | (подпись) | | |  | (фамилия, имя, отчество) | | |
|  | | | | |  |  | | |  |  | | |

ОГЛАВЛЕНИЕ

[1 Введение 4](#_Toc123070466)

[2 Теоретические основы и определение необходимых методов для создания NLP-системы 6](#_Toc123070467)

[3 Выбор библиотек для построения NLP конвейера 8](#_Toc123070468)

[4 Применение NLP в задачах нефтегазового комплекса 10](#_Toc123070469)

[5 Анализ результатов 12](#_Toc123070470)

[6 Выводы 14](#_Toc123070471)

[7 Список литературы 15](#_Toc123070472)

[Приложение А 16](#_Toc123070473)

# **1 Введение**

NLP — одно из направлений искусственного интеллекта, которое работает с анализом, пониманием и генерацией языков, для того, чтобы взаимодействовать с компьютерами устно и письменно, используя естественные языки вместо компьютерных. NLP представляет собой группу техник автоматической обработки естественного человеческого языка в формате устной речи или текста.

Поисковые системы — первая коммерчески успешная сфера приложения NLP. Поиск вдохновлял все более быструю разработку NLP-алгоритмов, которые затем усовершенствовали технологии поиска.

Некоторые области применения NLP:

* медицина (распознавание и анализ электронных медицинских карт пациентов);
* перевод текста;
* чат-боты;
* распознавание речи;
* поисковые системы;
* голосовые помощники.

Применяя методы NLP, можно решить или поспособствовать решению широкому спектру задач. Задачи можно поделить на несколько типов или уровней:

- уровень сигналов: распознавание текста, распознавание речи, синтез речи, синхронный перевод;

- уровень слов и словосочетаний: морфологический анализ слова, лемматизация, исправление грамматических ошибок, выявление отношений между словами, распознавание именованных сущностей;

- уровень предложений: синтаксический разбор предложений, генерация предложений, распознавание именованных сущностей, распознавание именных групп, анализ тональности;

- уровень документов: выявление ключевых слов, резюмирование текста, выделение заголовка, перевод текста, анализ тематики;

- уровень корпусов: устранение дубликатов, антиплагиат, информационный поиск.

Целью курсовой работы является применение NPL-системы для решения одной из задач нефтегазового комплекса, а именно обработка текста отчетов по производствам на НПЗ.

Для достижения цели выделены следующие этапы:

- определение необходимых методов для создания NLP-системы;

- выбор необходимых инструментов и библиотек для обработки естественного языка;

- реализация NLP-системы на языке Python;

- интерпретация и описание полученных результатов выполнения программы.

# **2 Теоретические основы и определение необходимых методов для создания NLP-системы**

2.1 Обработка текста

Обработка текста включает в себя несколько базовых методов и техник:

- токенизация по словам и/или предложениям;

- [лемматизация](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) и [стемминг](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BC%D0%BC%D0%B8%D0%BD%D0%B3) текста;

- удаление стоп-слов;

- регулярные выражения;

- техника извлечения признаков «[мешок слов](https://en.wikipedia.org/wiki/Bag-of-words_model)»;

- мера оценки [TF-IDF](https://ru.wikipedia.org/wiki/TF-IDF).

Токенизация (сегментация) по предложениям – это процесс разделения письменного языка на предложения-компоненты или слова-компоненты. Для выделения отдельных предложений используются знаки пунктуации, для слов разделителем могут служить пробелы. Токенизация также предполагает выделение из текста групп слов, называемых n-граммами.

Лемматизация и стемминг являются частными случаями нормализации и служат для приведения всех встречающиеся словоформ к одной, нормальной словарной форме. Лемматизация использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме. Стемминг отрезает «лишнее» от корня слов, часто это приводит к потере словообразовательных суффиксов.

Удаление стоп слов - избавление от общеупотребительных артиклей, местоимений и предлогов. В этом процессе некоторые очень распространенные слова, которые очевидно не представляют большой ценности для целей NLP, фильтруются и исключаются из обрабатываемого текста. Удаляются слова которые не несут информации о тексте.

Регулярные выражения (regexp) – последовательность символов, которая определяет шаблон для поиска символов. Используется для удаления символов из текста, не участвующих в анализе.

Техника «мешок слов» - техника извлечения признаков для построении модели, позволяющая подсчитывать все слова в фрагменте текста. Создает матрицу вхождений для предложения или целого документа, игнорируя грамматику и порядок слов. Эта частота появления или вхождения слов затем используются в качестве признаков для обучения классификатора.

TF-IDF (term frequency inverse document frequency). Частотность   
слова — количество вхождений слова в документ. Обратная частотность документа означает, что это количество для конкретного слова делится на число документов, в которых оно встречается. Представляется собой матрицу   
«терм — документ». Построенные векторы TF-IDF — показатели слов, наилучшим образом отражают степень их важности.

# **3 Выбор библиотек для построения NLP конвейера**

Основные наборы библиотек для обработки естественного языка:

- NTLK;

- spaCy;

- pymorphy2;

- Gensim;

- Scikit–learn;

- Natasha;

NTLK (Natural Language ToolKit) представляет собой пакет библиотек и программ для символьной и статистической обработки естественного языка. Содержит встроенный токенизатор слов и предложений, средства векторизации, обработки вероятностной информации. Не обладает высокой скоростью обработки информации. Имеет встроенную поддержку русского языка. Также в команде проекта работают разработчики из России. Есть средства визуализации.

SpaCy предназначена в основном для производственного использования. Поддерживаем механизмы глубокого обучения которые позволяют подключать статистические модели, обученные популярными библиотеками машинного обучения, такими как TensorFlow, PyTorch или MXNet, через собственную библиотеку машинного обучения Thinc. Есть средства визуализации. Нет встроенной поддержки обученных моделей на русском языке.

Pymorphy2 - морфологический анализатор, умеет приводить слово к нормальной форме, ставить слово в нужную форму (ставить во множественное число, менять падеж слова и т.п.), возвращать информацию о слове (число, род, падеж, часть речи и т.д.). Для работы использует словарь OpenCorpora. В настоящий момент проект заброшен и не обновляется. На версии Python 3.11 не запустился (методы, использованные в библиотеке Python посчитал устаревшими и выкинул исключение).

Gensim – это библиотека с открытым исходным кодом на Python, используется для тематического обучения без учителя и обработки естественного языка (NLP). Она предназначена для извлечения семантических тем из документов. Может работать с большими текстовыми коллекциями. В нее добавлены более удобные средства для обработки текста.

Scikit–learn – набор библиотек машинного обучения для языка программирования Python. Включает в себя различные алгоритмы классификации, регрессии и кластеризации, включая машины опорных векторов, случайные леса, градиентное усиление, k-means и DBSCAN. Подходит для обработки векторных представлений токенов слов и предложений, обучения нейронных сетей на их основе, к примеру, для решения задач классификации текста.

Natasha ­– открытая библиотека для языка программирования Python, которая позволяет извлекать структурированную информацию из текстов на русском языке. Включает в себя экстракторы для имён, адресов, сумм денег, дат и некоторых других сущностей. Решает задачи извлечения именованных сущностей (NER).

# **4 Применение NLP в задачах нефтегазового комплекса**

На предприятиях нефтегазового комплекса в настоящее время наиболее востребована автоматизация технологических процессов и производства. С ее помощью возможно:

- сокращение трудозатрат на обработку информации;

- сокращение издержек;

- снижение времени на выполнение технологических операций;

- снижение аварийных ситуаций;

- и прочее.

Для курсовой работы было выбрано направление по оптимизации обработки информации, а именно обработка текста документов на примере использующегося на одном из предприятий АО «Роснефть» отчета произошедших отключений блокировок противоаварийной защиты на производствах. Часть отчета представлена на рисунке А.1 в приложении А.

Для обработки текста необходимо составить NLP конвейер (pipeline), который также содержит выбранные основные этапы обработки естественного языка. Элементы конвейера будут располагаться в порядке, описанном в п.2.1. Схема конвейера представлена на рисунке 1.

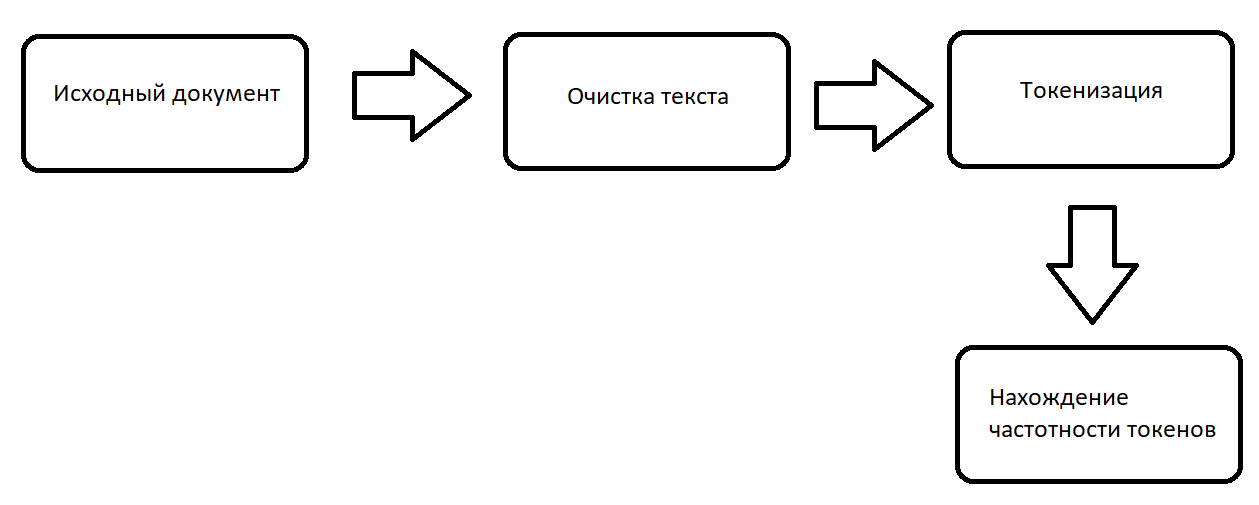


Рисунок 1 – NLP конвейер

Основным инструментом для построения конвейера был выбран NTLK, т.к. токенизаторы в его составе понимают русский язык, она простая и рекомендована для обучения. Также из NTLK потребуется класс FreqDist, который кодирует «частотные распределения», подсчитывающие, сколько раз встречается каждый результат эксперимента.

Для векторизации выбран scikit–learn, будет использован векторизатор CountVectorizer, который содержит в себе также токенизатор и фильтр стоп-слов.

Учитывая специфику работы библиотечных токенизаторов слов, которые в качестве отдельных токенов также берут цифры, символы и знаки препинания, часть важной информации при токенизации может потеряться. Важной информацией следует считать даты, наименования цехов, установок, технологических позиций, тегов, причины отключения блокировок и номера санкционирующих документов. Описания параметров и действий персонала связаны с позициями параметров и тегами, их также желательно не разделять на составляющие токены. Данные из датасета содержат имена тегов из базы данных временных рядов (используется для сбора, хранения, систематизации данных с датчиков оборудования) на латинице, символы и цифры

Для большего охвата информации из датасета были реализованы токенизаторы слов и предложений, учитывающие все вышенаписанное. Исходный код программы и результаты представлены в приложении А.

В качестве набора стоп-слов используется стандартный набор для русского языка из NTLK дополненный своим набором стоп-слов. В него входят слова и предложения, не имеющие информационной ценности в контексте обрабатываемого датасета.

# **5 Анализ результатов**

Результат работы программы представлен в виде кривой с наиболее часто встречающимися в тексте документа токенами (смотреть рисунки А.2, А.3) для первых 30 наиболее часто встречающихся в документе токенов.

На рисунке А.2 представлена кривая, построенная с помощью набора библиотек scikit–learn.

Кривая была построена на основе созданных токенов предложений с предварительно очищенными от пустых, «nan», цифр, дат, символов. Как видно из меток на кривой, стандартный токенизатор из scikit–learn разбил на токены значения тегов, описания параметров, действия и прочее.

Если сопоставить имена колонок с информацией из ячеек и полученными из нее токенами, можно выделить следующую суть документа:

- наиболее часто из читаемых слов встречается «насосы», соответственно можно сделать вывод о том, что отключения защиты в основном происходили

на установках, имеющих насосное оборудование;

- отключаемые установки имеют в своем составе измерители температуры и давления;

- чаще всего отключения защиты были на производстве топлива (ТП), производстве масел (ПМ), менее часто отключения происходили на нефтеперерабатывающем производстве (НПП) и производстве нефтехимии (ПНХ).

На рисунке А.3 представлена кривая, построенная с помощью реализованного под отчет простого токенизатора слов и использования метода для подсчета частотности слов в документе из библиотеки NTLK. Каждая метка на кривой - это отдельный неделимый токен, представляющий собой информацию из ячейки документа, где содержимое не разбито на другие токены.

На основании меток и частоты встречаемости токенов, а также результата сопоставления информации можно выделить следующую суть документа:

- «-нет-» означает, что большинство произошедших блокировок не имеют санкционирующего документа;

- как и на рисунке А.2, чаще всего отключения защиты были на производстве топлива (ТП), производстве масел (ПМ), менее часто отключения происходили на нефтеперерабатывающем производстве (НПП) и производстве нефтехимии (ПНХ);

**-** наибольшее количество отключений происходило в цехах № 1, 103;

- наибольшее количество отключений выявлено по установках 145/148, 1215, следовательно эти установки больше всего подвергалась ремонтам, ТО;

- также можно оценить в какие даты и время наиболее часто на установках проводились работы.

# **6 Выводы**

В результате обработки отчета и построения кривых частотного распределения токенов было установлено, что более полную информацию из отчета возможно получить используя специфичное разделение токенов.

При обработке стандартными методами из библиотек информативность получается меньше, однако лучше проанализирована описательная часть (столбцы описание параметра и описание алгоритма).

Для дальнейшего улучшения работы NLP конвейера необходимо будет объединить подход со специфичной обработкой основной части токенов и стандартные методы токенизации и обработки в описательной части отчета.

Итогом проделанной работы стало создание инструмента для получения обобщенной информации из отчета.

Полученные кривые позволяют управляющему персоналу, к примеру, главным технологам не просматривая всю информацию из отчета понять на какие объекты (цеха, установки и пр.) отключались чаще всего, выявить несанкционированные отключения противоаварийной защиты и пр.

# **7 Список литературы**

7.1 «Natural Language Processing in Action Understanding, analyzing, and generating text with Python» HOBSON LANE, COLE HOWARD, HANNES MAX HAPKE (перевод, 2020 г).

7.2 «Natural Language Processing with Python» Steven Bird, Ewan Klein, and Edward Loper.

7.3 Электронная документация к библиотеке NTLK

URL: https://www.nltk.org/api/nltk.html.

# **Приложение А**

А1 Исходный код программы на Python

**import** **nltk**

**import** **numpy** **as** **np**

**from** **matplotlib** **import** pyplot **as** plt

**from** **nltk.probability** **import** FreqDist

**from** **nltk.corpus** **import** stopwords

**from** **nltk** **import** BlanklineTokenizer

**import** **re**

**import** **pandas** **as** **pd**

**from** **sklearn.feature\_extraction.text** **import** CountVectorizer

**class** **ExcelParser**():

**def** **\_\_init\_\_**(self, excel\_name, file\_sheetname):

self.excel\_name = excel\_name

self.file\_sheetname = file\_sheetname

**def** **read\_excel**(self):

data\_frame = pd.read\_excel(self.excel\_name, sheet\_name=self.file\_sheetname)

**return** data\_frame.drop (columns=['Unnamed: 0','Unnamed: 25', 'Unnamed: 26'], index=[ **0**,**1**,**2**,**3**])

**class** **NLP**():

**def** **\_\_init\_\_**(self):

self.stop\_words = ['nan', ' ', '**\n**', 'тег отсутствует', 'Тег состояния технологического объекта не предусмотрен', 'на режиме', 'вкл', 'выкл','включена', 'отключена',

'тег состояния технологического объекта не предусмотрен', 'б/н']

self.stop\_words\_for\_sents = ['nan', '**\n**']

**def** **tokenize\_words**(self, words):

#предложения из ячеек тоже будем считать отдельными токенами, т.к. это описание и его менять нельзя

out\_words = []

**for** word **in** words:

**for** item **in** word:

tokens = str(item).lower()

out\_words.append(tokens)

out\_words = self.delete\_stopwords\_words(out\_words)

self.word\_tokens = out\_words

#датасет изначально уже практически токенизирован, просто делаем из листа единое предложение

**def** **tokenize\_sents**(self, sents):

out\_sents=[]

**for** sent **in** sents:

tokens= self.clean\_tokens(str(sent).lower())

out\_sents.append(tokens)

self.preprocessing\_sents(out\_sents)

self.sent\_tokens = out\_sents

**def** **clean\_tokens**(self, str):

replace\_symbols = ["[", "]", ",", "'", "{","}"]

out\_str = str

**for** sym **in** replace\_symbols:

out\_str = out\_str.replace(sym, '')

out\_str = out\_str.replace('**\\**n', ' ')

out\_str = out\_str.replace('**\n**', ' ')

**return** out\_str

**def** **delete\_stopwords\_words**(self, tokens):

without\_stopwords = [token **for** token **in** tokens **if** token **not** **in** self.stop\_words]

**return** without\_stopwords

**def** **preprocessing\_sents**(self, tokens):

cleanr = re.compile('[!@"“’«»#$%&**\'**()\*+,—/:;<=>?^\_`{|}~\[\]]')

**for** i, token **in** enumerate(tokens):

**for** sym **in** self.stop\_words\_for\_sents:

tmp\_str = tokens[i].replace(sym, '')

cleantext = re.sub(cleanr, '', tmp\_str)

rem\_num = re.sub('[0-9]+', '', cleantext)

tokens[i] = rem\_num

**def** **vectorize**(self, tokens\_sent, most\_count):

new\_custom\_stop\_words = ['поз', 'режиме', 'тег', 'anhk','2022', 'раздел', 'datetime', 'time', '2023', 'blok', 'uv', 'ob',

'другой', 'ей', 'какоий', 'мои', 'ней', 'объекта', 'отсутствует', 'предусмотрен', 'сейчас', 'состояния', 'такой', 'технологического', 'этой']

stop\_words\_new = stopwords.words('russian')

stop\_words\_new.extend(new\_custom\_stop\_words)

stop\_words\_new.extend(self.stop\_words)

vecorize = CountVectorizer(strip\_accents='unicode', stop\_words=stop\_words\_new)

words = vecorize.fit\_transform(tokens\_sent)

word\_frequency = pd.DataFrame(

{'word' : vecorize.get\_feature\_names\_out(),

# получаем частотность слов

# находя сумму компонент векторов

'frequency' : np.array(words.sum(axis = **0**))[**0**]

}).sort\_values(by = 'frequency', ascending = **False**)

**return** word\_frequency[:most\_count]

**class** **Pipeline**:

**def** **\_\_init\_\_**(self, excel\_name, sheetname):

self.excel\_name = excel\_name

self.excel\_sheetname = sheetname

**def** **show\_most\_common**(self, most\_count):

ew = ExcelParser(self.excel\_name, self.excel\_sheetname)

work\_data = ew.read\_excel()

nlp = NLP()

data = work\_data.values.tolist()

nlp.tokenize\_words(data)

fdist = FreqDist(nlp.word\_tokens)

fdist.most\_common(most\_count)

fdist.plot(most\_count, cumulative=**False**)

**def** **show\_most\_common\_vectorized**(self, most\_count):

ew = ExcelParser(self.excel\_name, self.excel\_sheetname)

work\_data = ew.read\_excel()

nlp = NLP()

data = work\_data.values.tolist()

nlp.tokenize\_sents(data)

word\_frequency\_filtered = nlp.vectorize(nlp.sent\_tokens, most\_count)

fig, ax = plt.subplots()

ax.plot(word\_frequency\_filtered['word'], word\_frequency\_filtered['frequency'])

ax.grid()

plt.xticks(rotation=**90**)

plt.show()

**def** **main**():

nltk.download('punkt')

nltk.download('stopwords')

pipe = Pipeline("test\_dataset.xlsx", "Data")

pipe.show\_most\_common(**30**)

pipe.show\_most\_common\_vectorized(**30**)

**if** \_\_name\_\_=="\_\_main\_\_":

main()

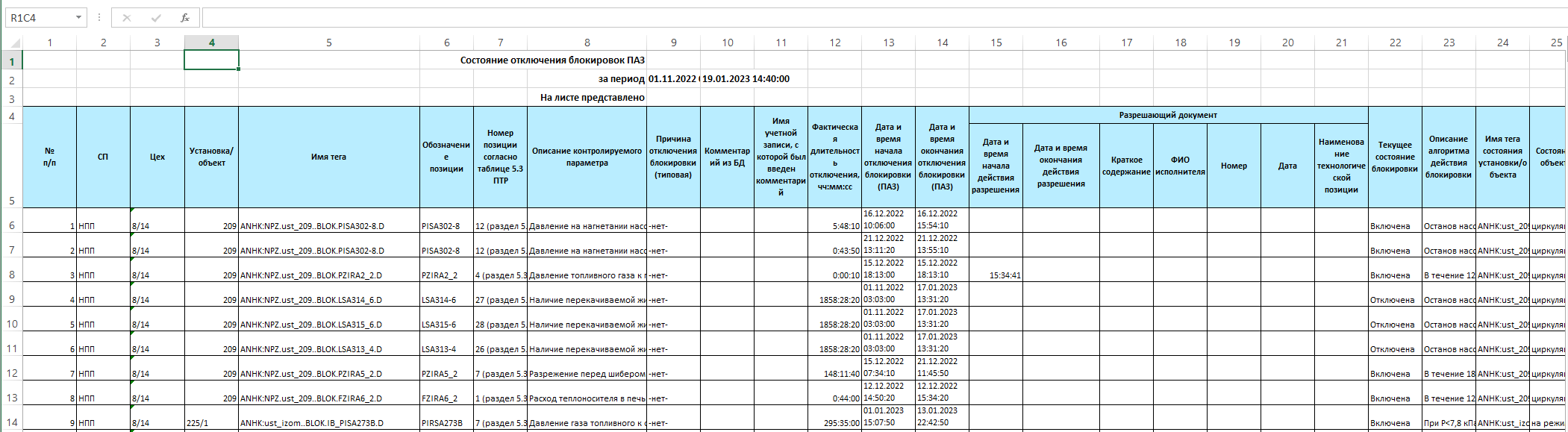
А2 Небольшая часть данных из исходного отчета по производствам представлена на рисунке А.1. 

Рисунок А.1 – Общий вид отчета

А3 Кривая значимости слов в тексте c использованием sklearn представлена на рисунке А.2

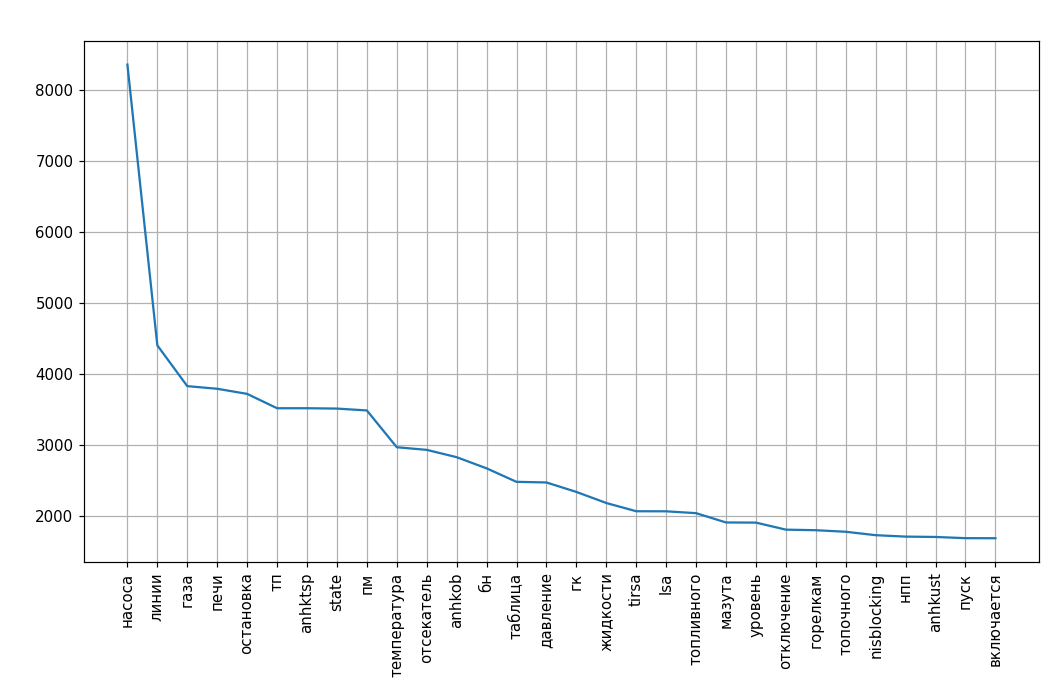


Рисунок А.2

А4 Кривая значимости слов в тексте c использованием своей реализации токенизаторов представлена на рисунке А.3

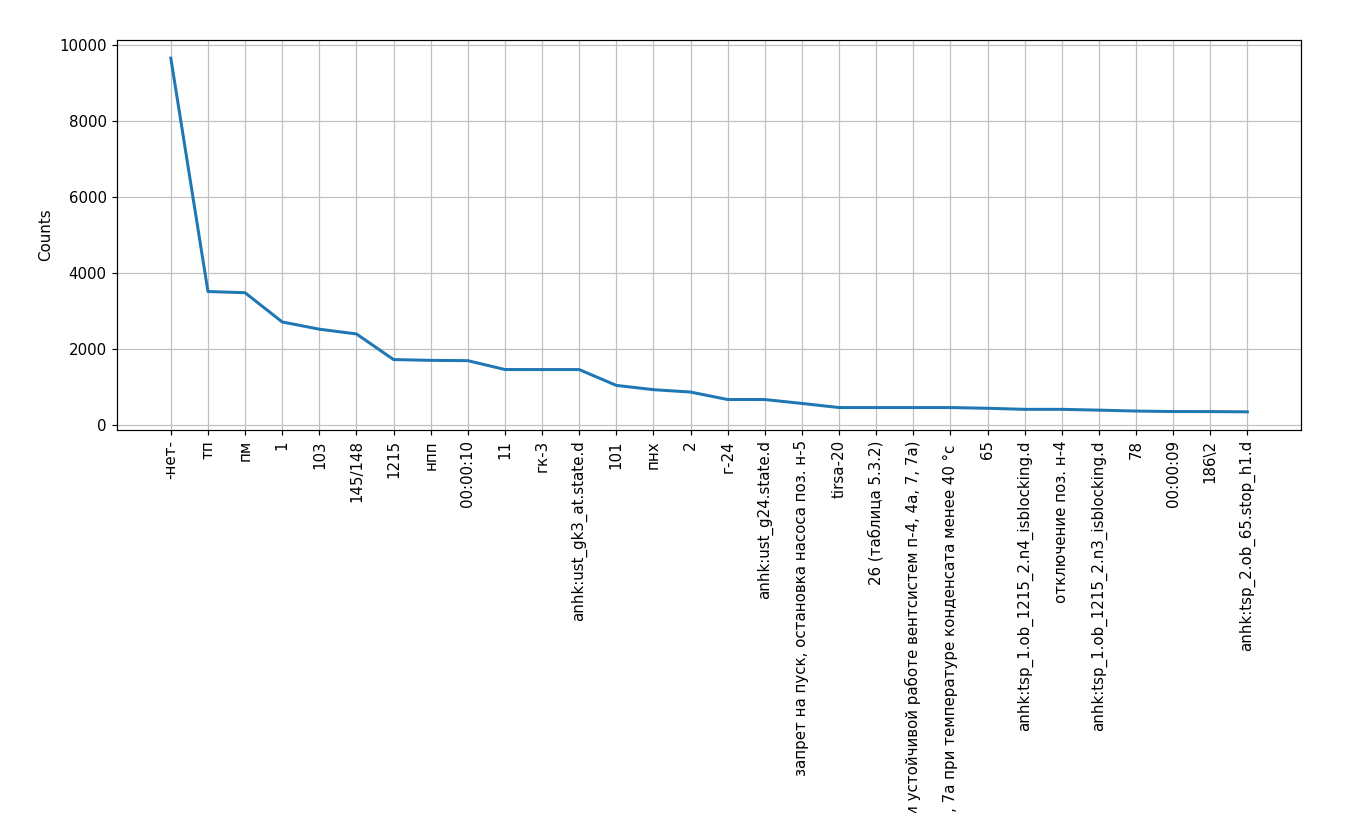


Рисунок А.3