Санкт–Петербургский государственный университет

ЗАМЯТИН Павел Степанович

Выпускная квалификационная работа

Автоматическая генерация временных меток для видео на платформе Youtube

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика» Основная образовательная программа СВ.5005.2017 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
доцент, кафедра технологии программирования,
кандидат физ.-мат. наук, Овсянников Александр Дмитриевич
Рецензент:
доцент, кафедра теории систем управления электрофизической аппаратурой, кандидат физ.-мат. наук, Головкина Анна Геннадьевна

Санкт-Петербург
2021

Содержание

Введение

Актуальность

YouTube [1] — видеохостинг, предоставляющий пользователям услуги хранения, доставки и показа видео. YouTube стал популярнейшим видеохостингом и вторым сайтом в мире по количеству посетителей.

Согласно некоторым источникам каждую минуту на платформу YouTube загружается более 500 часов видеоконтента [2].

Чтобы пользователи могли ориентироваться в таком огромном количестве видеоконтента, разработчики из компании Google постоянно расширяют функционал платформы, добавляя новые инструменты [3]. Так любой видеоролик можно разделить на главы (chapters), которые в значительной мере облегчают навигацию. Но для работы данной функции необходимо наличие под видео специальных временных меток, которые бы указывали на начала отдельных глав.

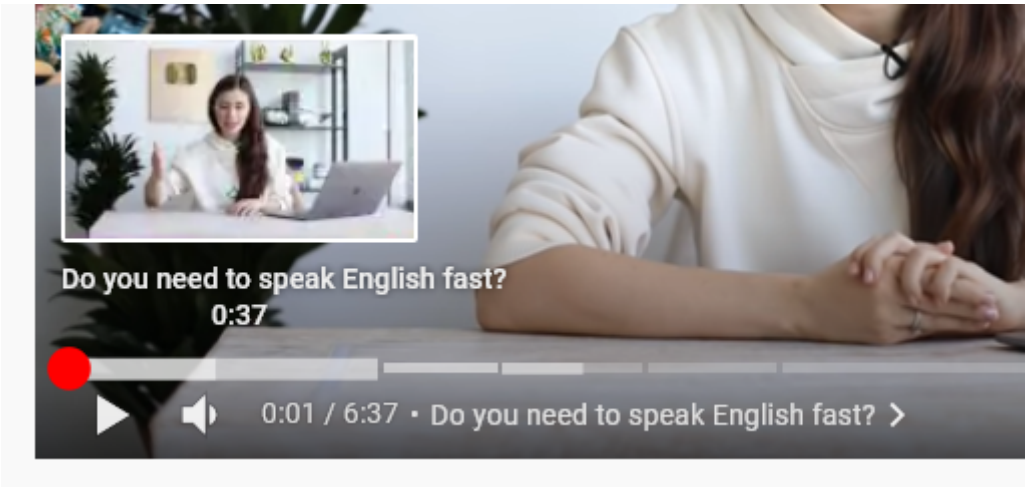

Рис. 1. Видеоролик, разделенный на главы.

Стоит отметить, что авторы редко добавляют временные метки для своих видео. В некоторых случаях зрители сами могут оставить список меток в комментариях.

Так как зачастую временные метки просто отсутствуют, то возникает идея сделать процесс создания временных меток автоматическим.

Рис.2. Временные метки, оставленные зрителями.

Кому может быть полезно решение задачи по автоматической генерации временных меток для видео с платформы YouTube? Во-первых, для компании Google. Поисковая система от компании Google уже сейчас может выдавать в качестве результата поиска ссылка на ключевой фрагмент видео. Но если у видеоролика отсутствуют временный метки, то поисковик не сможет ссылаться на нужный момент.

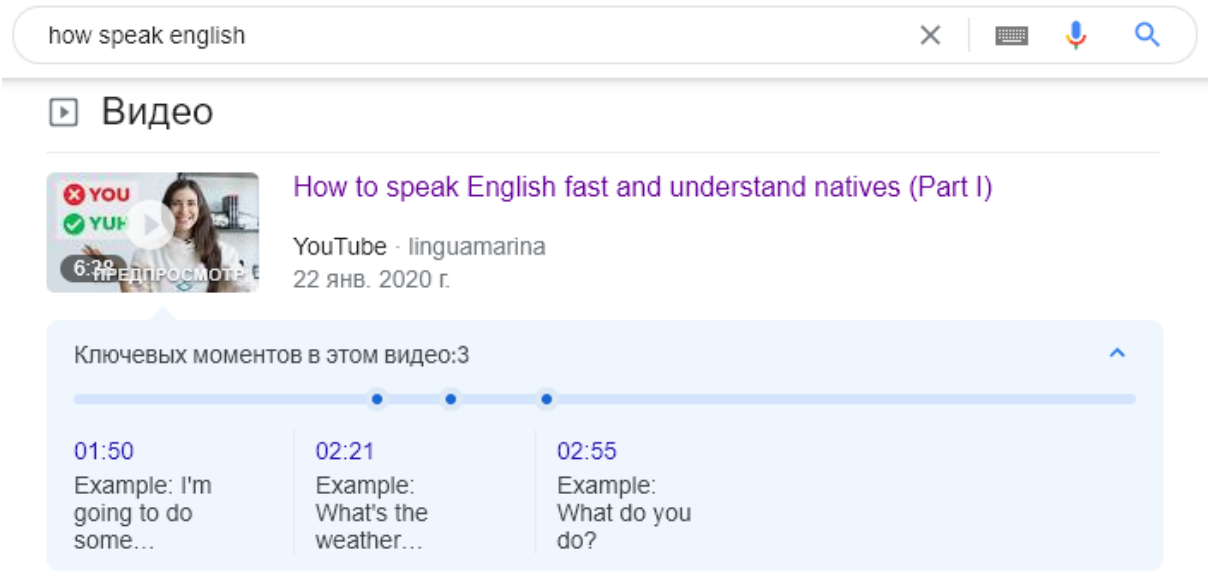

Рис. 3. Результаты поиска в Google.

Во-вторых, для создателей видео. Автоматическая генерация временных меток значительно упростит им работу. При наличии навигации

видео станет более понятным, следовательно зрители с большей вероятностью будут смотреть данное видео, а не искать другое.

В-третьих, для зрителей, ведь смотреть видео с навигацией значительно удобнее.

Постановка задачи

Дано видео с платформы YouTube с субтитрами на английском языке. Субтитры могут быть как написаны вручную, так и сгенерированы автоматически с помощью встроенных алгоритмов YouTube.

Необходимо получить список, состоящий из временных меток, указывающих на начало отдельных глав (смысловых частей), и краткого описания глав.

Обзор существующих решений

На данный момент компания Google ведет закрытое тестирование алгоритма, решающего данную задачу [3].

Какими методами они решают задачу не известно. Известно только то, что разработчики используют алгоритмы машинного обучения для распознавания текста и автоматического добавления глав в видео. Каких-либо других готовых решений поставленной задачи на данный момент нет.

Можно разбить поставленную задачу автоматической генерации временных меток на подзадачи:

1. Извлечение субтитров из видео.
2. Сегментация текста на предложения.
3. Сегментация текста на главы.
4. Генерация краткого описания глав.
5. Сопоставление начал глав с временной шкалой.

Цель дипломной работы

Целью работы является разработка программного комплекса на языке Python, решающего поставленную задачу.

Задачи дипломной работы

1. Рассмотреть существующие подходы к решению подзадач 2-4.
2. Разобрать принцип работы реализованных алгоритмов, подходящих для решения подзадач.
3. Провести сравнение эффективности алгоритмов библиотек для подзадачи 2.
4. Разработать программный комплекс, решающий поставленную задачу.

Глава 1. Общий подход к решению

Ключевой идеей работы является то, что можно работать не с видео целиком, а только с субтитрами, написанными вручную или полученными после работы встроенных алгоритмов YouTube по распознаванию речи. Субтитры представляют с собой текст, к которому можно применять различные методы по обработке естественного языка (англ. Natural Language Processing, NLP) [4].

Можно сформулировать краткое описание решения поставленной задачи. С помощью готовых библиотек из видео извлекаются субтитры. Если в тексте субтитров отсутствует пунктуация, то он разбивается на предложения. Текст с пунктуацией разбивается на главы, содержащие связанные по смыслу предложения. Для каждой главы генерируется краткое описание. В конце начала каждой главы сопоставляется с временной шкалой. В результате получается список временных меток.

Глава 2. Обзор существующих библиотек

2.1 Извлечение субтитров из видео.

Для начала работы необходимо извлечь из исходного видео субтитры. Субтитры можно использовать в качестве исходного текста. Для этой задачи воспользуемся готовой библиотекой.

YouTubeTranscriptApi [5] - python API, позволяющий получить субтитры для видео с платформы YouTube. Он хорошо работает работает как для автоматически сгенерированных субтитров, так и для субтитров, написанных вручную. На вход подается ссылка на видео, а результатом работы является список словарей. Каждый словарь состоит из текста субтитра, временной метки, указывающей на начало субтитра, и его продолжительности.

```
[
    {
        'text': 'Hey there',
        'start': 7.58,
        'duration': 6.13
    },
    {
        'text': 'how are you',
        'start': 14.08,
        'duration': 7.58
    },
    # ...
]
```

Рис. 4. Пример результата работы YouTubeTranscriptApi.

2.2 Сегментация текста на предложения.

Если авторы видео не загрузили вручную написанный текст субтитров, то исходный текст, полученный на прошлом этапе был сгенерирован с помощью системы автоматического распознавания речи YouTube. Такой текст не содержит какой-либо пунктуации, он трудно читаемый, и его невозможно

использовать для дальнейшей работы алгоритмов обработки естественного языка.

Существует ряд методов, решающих подзадачи по восстановлению знаков препинания, в которых используются различные подходы: n-граммовые модели [6], условные случайные поля (CRF) [7, 8], анализ зависимостей на основе переходов [9], глубокие и сверточные нейронные сети [10]. Некоторые имеют рассматривал восстановление знаков препинания как задачу машинного перевода, перевод текста без знаков препинания в текст с пунктуацией [11, 12].

Подробнее рассмотрим принцип работы готовых библиотек DeepSegment [13], NNsplit [14], Punctuator [15]. Алгоритмы данных библиотек основываются на рекуррентных нейронных сетей (RNN), ниже кратко описаны используемые архитектуры RNN.

2.2.1 Рекуррентные нейронные сети

Рекуррентная нейронная сеть (англ. Recurrent Neural Network, RNN)[16] - нейронная сеть, которая хороша для моделирования последовательных данных. На каждом этапе работы сети внутренний слой нейронов получает набор входных данных X и информацию о предыдущем состоянии внутреннего слоя h, на основе которых вычисляется выход сети y.

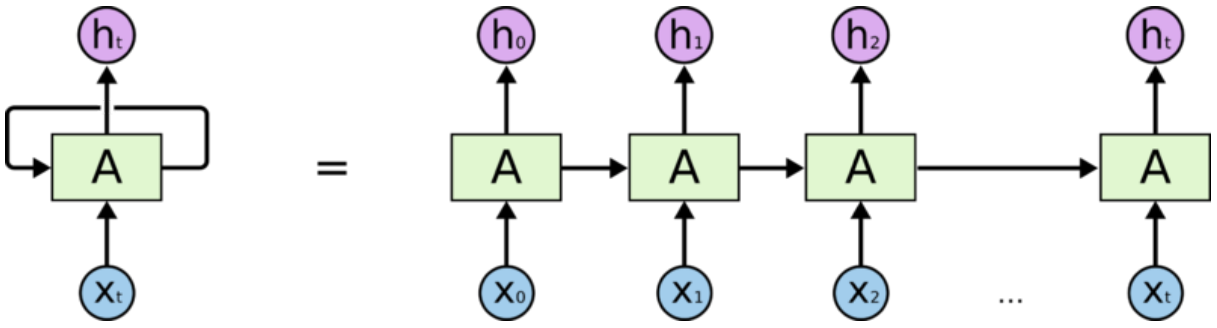

Рис. 5. RNN и ее развернутое представление

Долгая краткосрочная память (англ. Long short-term memory, LSTM) [17] — разновидность рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. LSTM-модули разработаны специально, чтобы избежать проблемы долговременной зависимости, запоминая значения как на короткие, так и на длинные промежутки времени.

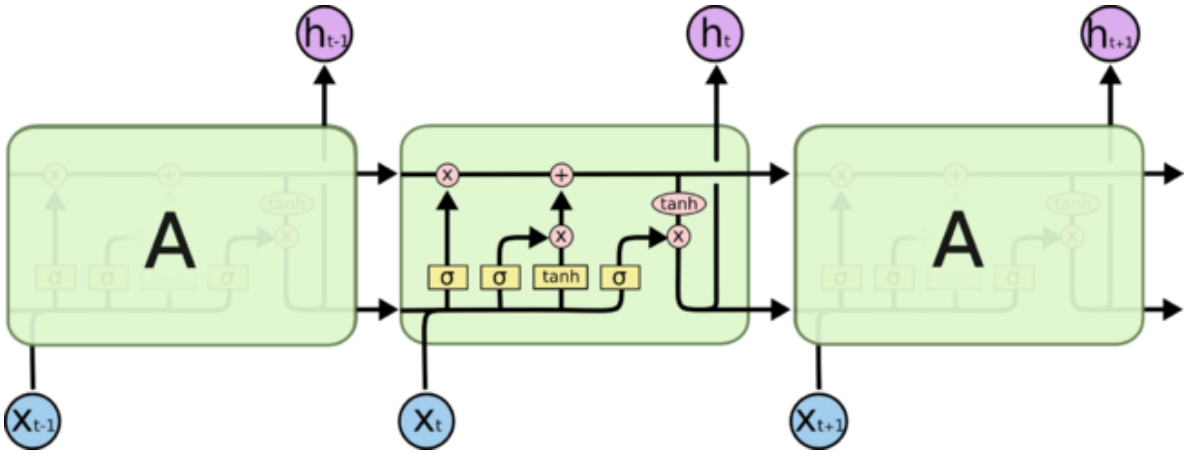

Рис. 6. Схема слоев долго-краткосрочной памяти.

Двунаправленная рекуррентная сеть (англ. Bidirectional Recurrent Neural Network, biRNN/ BRNN) [18] - состоит из двух однонаправленных рекуррентных сетей, которые обрабатывают входные данные различных направлениях, одна в прямом, другая - в обратном. Для каждого элемента входной последовательности X считается два вектора скрытых состояний A и A', на основе которых вычисляется выход сети y. Благодаря данной архитектуре сети доступна информация о контексте как из прошлого, так и из будущего.

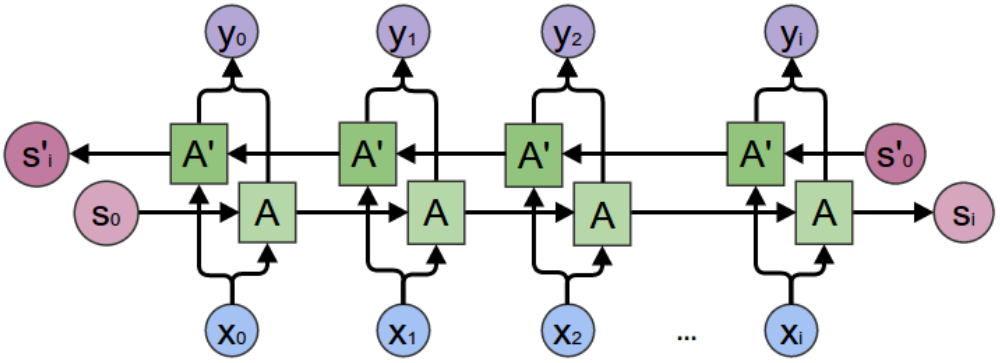

Рисунок 7. Двунаправленная рекуррентная сеть.

LSTM - CRF [19] - архитектура рекуррентной нейронной сети, которая является комбинацией LSTM и CRF (Conditional Random Field) сетей. Данная модель может эффективно использовать входные значение LSTM слоя и информацию тегов на уровне предложений CRF слоя. Слой CRF представлен линиями, соединяющими последовательные выходные уровни.

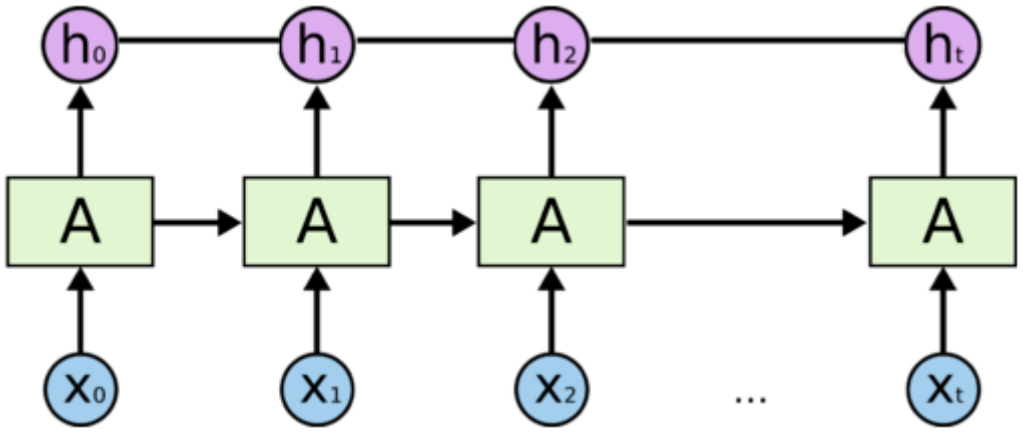

Рис. 8. Модель LSTM-CRF.

Управляемые рекуррентные блоки (англ. Gated Recurrent Units, GRU) – архитектура рекуррентной нейронной сети [20]. Было установлено ее эффективность при решении задач моделирования музыкальных и речевых сигналов сопоставима с использованием LSTM [21] По сравнению с LSTM у данного механизма меньше параметров.

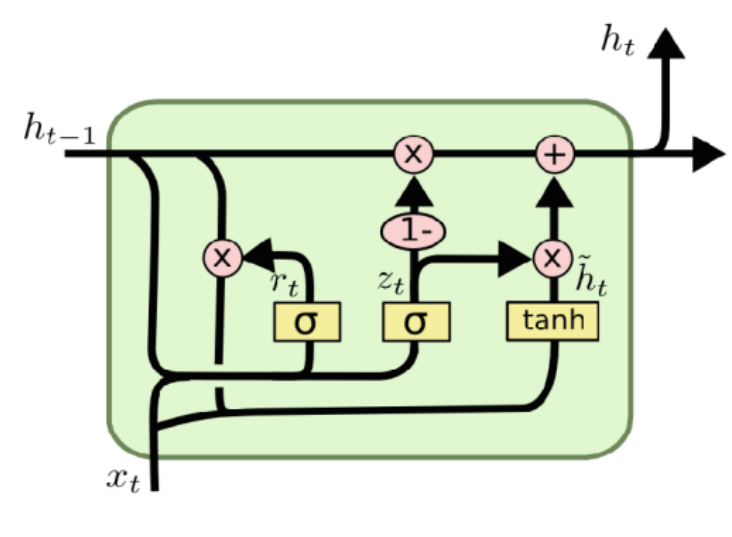

Рис. 9. Схема GRU слоя.

2.2.2 NNSplit

NNSplit [14] - библиотека, решающая задачу сегментации текста с помощью нейронных сетей. Основное приложение - определение границ предложения, но также поддерживается составное разделение для немецкого языка.

Алгоритм работы:

1) Абзацы извлекаются из текстов.

```
[
"Paragraph with multiple sentences. Some more text.",
        "Multiple million sentences here.",
                    ...
                    ]
```

Рис. 10. Извлечение абзацев из текста.

2) Абзацы разбиваются на слова и предложения, с помощью SoMaJo[22] (токенайзера для английских и немецких текстов).

```
                        [
[["Paragraph", "with", "multiple "sentences", "."],
          ["Some", "more", "text", "."]],
[["Multiple", "million", "sentences", "here", "."]],
                       ...
                        ]
```

Рис. 11. Токенизация.

3) Добавление ошибок. С некоторой вероятностью слова в начале предложения преобразуются из верхнего регистра в нижний, а точки в конце предложения удаляются. Это шаг, который позволяет NNSplit быть более терпимым к ошибкам.

```
                        [
[["Paragraph", "with", "multiple "sentences"],
        ["some", "more", "text", "."]],
[["multiple", "million", "sentences", "here"]],
                       ...
                        ]
```

Рис. 12. Добавление ошибок.

4) Из текста извлекаются фрагменты фиксированной длины. Это делает NNSplit инвариантным относительно длины исходного текста.

5) Модель LSTM учится предсказывать две метки для каждого символа (конец слова и конец предложения).

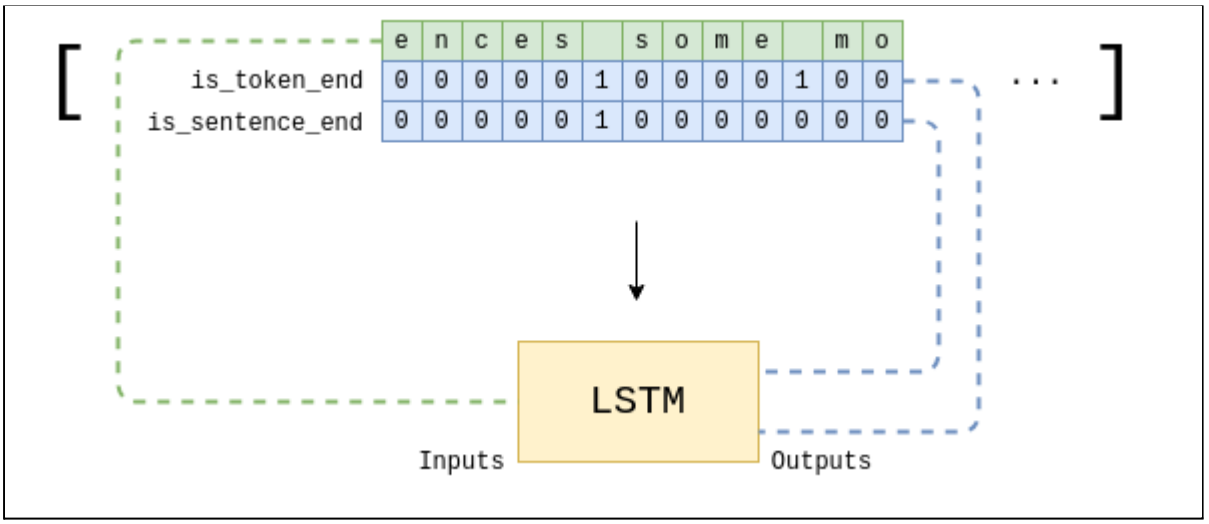

Рис. 13. Обучение модели.

6) Обученная модель делает прогнозы по каждой части исходного текста.

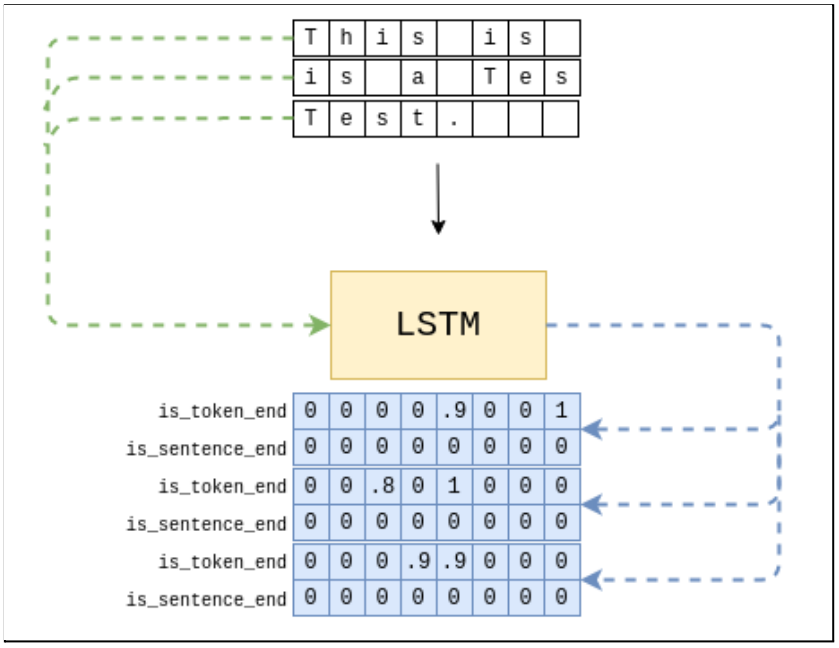

Рис. 14. Применение модели.

7) Затем результаты усредняются и на выходе мы получаем сегментированный текст.

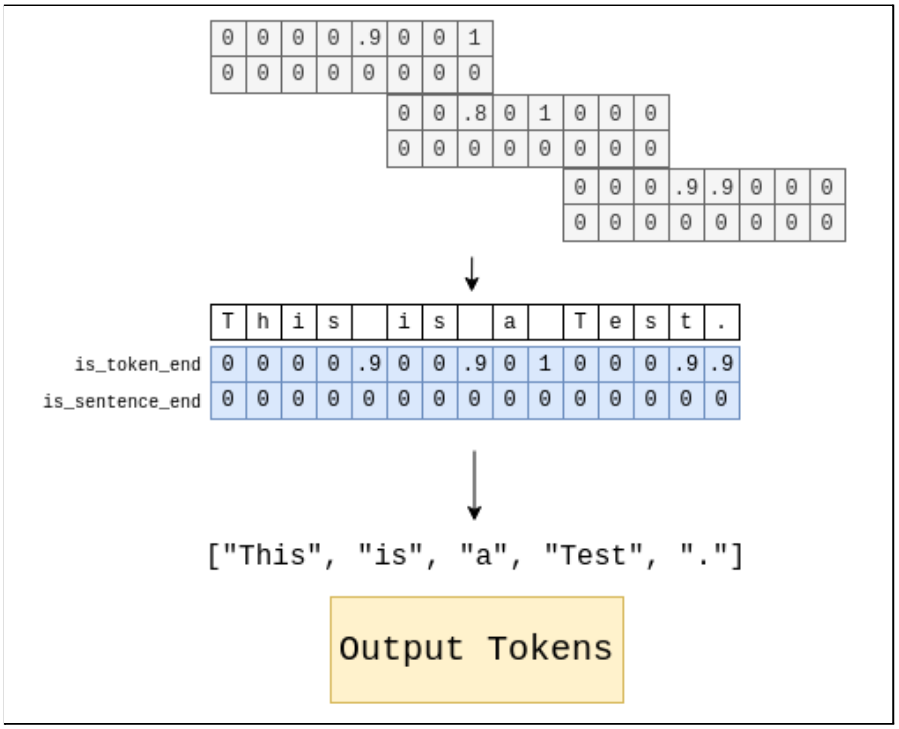

Рис. 15. Получение сегментированного текста.

Для обучения NNSplit использует базу данных Linguatools, содержащую тексты из Википедии и информацию о них: само содержание статьи, границы абзацев, количество ссылок на статью, ссылки на доступные переводы на другие языки, категории и многое другое.

2.2.3 DeepSegment

DeepSegment [13] - библиотека, решающая задачу сегментации текста. Она значительно превосходит стандартные библиотеки (spacy, nltk, corenlp).

DeepSegment использует модель BiLSTM-CRF для маркировки предложений (https://github.com/bedapudi6788/seqtag) вместе с предварительно обученным векторным представлением слов (word embedding).

Для обучения использовались автоматически сгенерированные предложения с помощью tatoeba https://tatoeba.org/ru . Предложения объединялись случайным образом, а модель училась предсказывать начала предложений.

2.2.4 Punctuator

Punctuator [15] - библиотека реализующая модель двунаправленной рекуррентной нейронной сети с механизмом внимания для восстановления пропущенной пунктуации между словами в несегментированном тексте.

Модель представляет собой двунаправленную рекуррентную нейронную сеть (BRNN), которая позволяет использовать контексты нефиксированной длины предыдущих и последующих позиции в тексте. В рекуррентных слоях используются управляемые рекуррентные блоки (GRU), которые хорошо подходят для фиксации зависимостей на большом расстоянии. GRU имеют те же преимущества, что и модули LSTM, но при

этом более просты. В модель включен механизм внимания [23], который повышает способность находить релевантные части контекста для принятия решений о пунктуации. Чтобы объединить воедино состояние модели в текущем входном слове и выходе из механизма внимания, используется подход позднего объединения [24], адаптированный из LSTM в GRU. Это позволяет выходным данным модели внимания напрямую взаимодействовать с текущим состоянием слоя, не влияя на его память.

Модели реализованы с использованием Theano и обучены на графических процессорах.

2.2.5 Сравнение эффективности

Метрики

Для оценки эффективности работы алгоритмов воспользуемся следующими метриками.

- $precision = \frac{TP}{TP + FP}$;
- $recall = \frac{TP}{TP + FN}$;
- $f1 - score = 2 \bullet \frac{precision \bullet recall}{precision + recall}$, где
- TP (true positive) — количество меток, верно соотнесенных категории;
- FP (false positive) — количество меток, неверно соотнесенных категории;
- FN (false negative) — количество меток, неверно не соотнесенных категории.

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного

класса из всех объектов положительного класса нашел алгоритм. Для нашей задачи объектом является граница между предложениями.

Данные

Для проведения экспериментов воспользуемся базой данных текстов слушаний Европейского парламента [25]. База данных содержит около 10000 текстов на английском языке, средняя длина текста около 5700.

```
<CHAPTER ID=1>
Resumption of the session
<SPEAKER ID=1 NAME="President">
I declare resumed the session of the European Parliament adjourned on Fr
iday 17 December 1999, and I would like once again to wish you a happy n
ew year in the hope that you enjoyed a pleasant festive period.
<P>
Although, as you will have seen, the dreaded 'millennium bug' failed to
materialise, still the people in a number o...
```

Рис. 16. Пример начала исходного текста.

Исходные тексты проходят обработку: удаляются фрагменты в треугольных скобках, все символы переводятся в нижний регистр, удаляются знаки препинания, кроме точек.

```
 i declare resumed the session of the european parliament adjourned on f
riday 17 december 1999 and i would like once again to wish you a happy n
ew year in the hope that you enjoyed a pleasant festive period.  althoug
h as you will have seen the dreaded millennium bug failed to materialise
still the people in a number o...
```

Рис. 17. Пример начала обработанного текста.

Полученные текст будут являться контрольными, с ними будут сравнивать результаты работы алгоритмов.

Сравнение эффективности.

Из контрольных текстов удаляются точки. Алгоритмы сегментации обрабатывают каждый текст, затем результаты работы сравниваются с контрольными текстами, на основе чего считаются TP, FP, FN. В конце для каждого алгоритма считаются средние показатели precision, recall и f1-score.

	precision	recall	f1-score
DeepSegment	0.54	0.53	0.53
NNsplit	0.54	0.77	**0.63**
punctuator	0.35	0.92	0.50

Таблица 1. Сравнение эффективности

Алгоритм NNsplit показал себя лучше других на данном наборе текстов (таблица 1, f1-score), поэтому сделаем его стандартным при вызове функций программного комплекса. Но оставим за пользователем возможность вызова двух других алгоритмов.

Также стоит отметить скорость работы NNsplit по сравнению с другими алгоритмами.

2.3 Сегментация текста на главы.

Изучив мета анализ статей по сегментации текста [26], можно сделать вывод, что алгоритмов разбиения текста, подходящих для нашей задачи, существует не так много. У большинства существующих методов нет реализации, либо необходимо потратить значительное количество времени на обучении моделей. [27]

Поэтому подробно рассмотрим работу реализованного алгоритма под названием TextTiling[28], идеально подходящего для решения данной подзадачи.

Алгоритм TextTiling реализован в пакете библиотек NLTK. Данный алгоритм обнаруживает сдвиги подтем на основе анализа лексических паттернов совместной встречаемости.

Процесс начинается с токенизации текста в псевдопредложения фиксированного размера w. Затем, в зависимости от используемого метода,

баллы схожести присваиваются в промежутках между предложениями. Алгоритм определяет разницу пиков между этими оценками и отмечает их как границы. Границы нормализуются до ближайшего разрыва абзаца, затем возвращается сегментированный текст.

Алгоритм состоит из трех этапов:

1. Токенизация (Tokenization)
2. Определение лексической оценки (Lexical Score Determination)
3. Идентификация границ (Boundary Identification)

2.3.1 Токенизация

К токенизации относится разделение исходного текста на отдельные лексические единицы - токены (token), в нашем случае это слова. Все символы преобразуются в нижний регистр. Каждый токен проверяется на вхождение в списки стоп-слов и часто встречающихся слов, если токен входит в один из этих списков, то он отбрасывается, в противном случае сокращается до своего корня с помощью функции морфологического анализа.

Затем текст разделяется на псевдопредложения заранее определенного размера w (параметр алгоритма), а не на «настоящие» синтаксически определенные предложения. Это сделано для того, чтобы можно было сравнивать единицы равного размера, поскольку количество общих терминов между двумя длинными предложениями и между длинным и коротким предложением, вероятно, дало бы несравнимые оценки.

Группы токенов будем называть последовательностями токенов (token-sequences). Морфологически проанализированный токен сохраняется в таблице вместе с записью порядкового номера последовательности токенов, в котором он появился, и количества раз, когда он появлялся в последовательности токенов. Стоп-слова участвуют в вычислении размера

последовательности токенов, но не в вычислении сходства между блоками текста.

2.3.2 Определение лексической оценки

В алгоритме используются два метода для определения оценки, которая будет назначена для каждого разрыва. Первый - сравнение блоков. Сравниваются соседние блоки текста, чтобы увидеть, насколько они похожи в зависимости от количества общих слов. Второй - метод ведения словарного запаса, каждому разрыву присваивается балл на основе того, сколько новых слов было замечено в интервале, в котором он является средней точкой.

а) Блоки

В алгоритме сравнения блоков смежные пары блоков последовательностей токенов сравниваются на предмет общего лексического сходства. Размер блока, обозначенный k (приблизительно соответствует средней длине абзаца), представляет собой количество последовательностей токенов, которые сгруппированы вместе в блок для сравнения с соседней группой последовательностей токенов.

Значения подобия вычисляются для каждого номера разрыва; то есть балл присваивается промежутку i, соответствующему тому, насколько похожи последовательности токенов от i - k до i к последовательностям токенов от i + 1 до i + k + 1. Лексическая оценка схожести между блоками вычисляется следующим образом.

$$score(i) = \frac{\sum_t w_{t,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}}$$

$$b_1 = \{token\text{-}sequence_{i-k}, \ldots, token\text{-}sequence_i\}$$

$$b_2 = \{token\text{-}sequence_{i+1}, \ldots, token\text{-}sequence_{i+k+1}\},$$

Рис.18 формулы

где t варьируется по всем токенам, а W - вес (частота), присвоенный терму t в блоке b. Эта формула дает оценку от 0 до 1 включительно.

b) Словарь

Лексическая оценка представляет собой отношение новых слов в интервале, деленное на длину этого интервала.

Токенизация осуществляется, как описано выше, с исключением стоп-слов и выполнения морфологического анализа. Затем балл присваивается разрыву следующим образом: количество еще не увиденных слов в последовательности токенов слева от разрыва добавляется к количеству еще не увиденных слов в лексеме последовательность справа, и это число делится на общее количество токенов в двух последовательностях токенов, или w * 2. w установлено равным 20, это дает длину интервала 40, что близко к параметру 35, как наиболее актуальному (Youmans 1991).

Лексическая оценка рассчитывается следующим образом. Для каждого промежутка i в последовательности лексем создается интервал b длиной w * 2 (где w - длина последовательностей лексем) с центром вокруг i, и b разделяется на две части равные части - bl и b2.

$$score(i) = \frac{NumNewTerms(b_1) + NumNewTerms(b_2)}{w * 2}$$

$$b_1 = \{token\text{-}sequence_{i-k}, \ldots, token\text{-}sequence_i\}$$
$$b_2 = \{token\text{-}sequence_{i+1}, \ldots, token\text{-}sequence_{i+k+1}\},$$

Рис.19. формулы

где NumNewTerms возвращает число новых термов в заданном интервале. Как и в блочной версии алгоритма, оценка отображается на

промежутке между последовательностями токенов и может варьироваться от 0 до 1 включительно.

2.3.3 Идентификация границ

Идентификация границ выполняется одинаково для всех методов лексической оценки и присваивает оценку глубины (если таковая имеется) для каждого разрыва. Оценка глубины соответствует тому, насколько сильно произошли изменения по обе стороны от данного разрыва, и основана на расстоянии до ближайших пиков по обе стороны.

Более формально, для данного разрыва i программа ищет разрыв L слева от i такой, что оценка для L - 1 была бы меньше, чем оценка для L. Точно так же для разрыва справа от i, оценка для K + 1 будет меньше, чем оценка для K. Результатом оценка глубины в i будет сумма score(R) - score(i) и score(L) - score(i). Например для графика (a) оценка в a_2 будет (Y_a_1 - Y_a_2) + (Y_a_3 - Y_a_2).

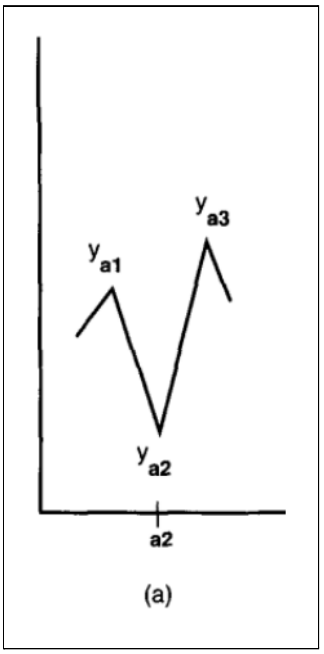

Рис.20. График.

Оценки глубины используются для определения границ сегмента. Чем больше оценка, тем больше вероятность того, что граница встречается в этом

месте. Выполняется предварительная проверка для предотвращения присвоения очень близких границ соседних сегментов. Между границами требуется как минимум три промежуточных последовательности токенов. 3 Это помогает контролировать тот факт, что многие тексты содержат ложную информацию заголовка и абзацы, состоящие из одного предложения.

2.4 Генерация краткого описания глав.

Экстрактивная суммаризация - извлечения из исходного текста наиболее «значимых» информационных блоков. В качестве блока могут выступать отдельные абзацы, предложения или ключевые слова. Экстрактивная суммаризация плохо подходит для поставленной задачи, так как отдельные предложения плохо передают основную мысль главы.

Абстрактивная суммаризация - генерация краткого содержания с порождением нового текста, содержательно обобщающего первичный документ. Абстрактивная суммаризация требует больших трудозатрат при реализации алгоритма. [29]

Существуют алгоритмы, которые решают задачу по извлечению ключевых слов из текста. Для исходного текста алгоритм идентифицирует набор терминов, которые лучше всего его описывают. Результат работы данных алгоритмов отлично подходят для нашей подзадачи. Рассмотрим существующие алгоритмы решения. [30]

TextRank [31] - это неконтролируемый (unsupervised) метод извлечения ключевых слов и предложений. Он основан на графе, где каждый узел представляет собой слово, а ребра строятся путем наблюдения совместного появления слов внутри движущегося окна предопределенного размера. Важные узлы графика, вычисленные с помощью алгоритма, подобного PageRank[32], представляют ключевые слова в тексте.

TopicRank - еще один неконтролируемый экстрактор ключевых фраз на основе графа [33]. В отличие от TextRank, в этом случае узлами графа являются темы, а каждая тема представляет собой кластер одинаковых однословных и многословных выражений.

BERT (Bidirectional Encoder Representations from Transformers) - это основанная на трансформере модель [34] для обработки естественного языка. Предварительно обученные модели могут преобразовывать предложения или слова в языковое представление, состоящее из массива чисел (embedding). Предложения или слова, имеющие похожие скрытые представления (embedding), должны иметь похожие семантические значения. Реализация, использующая этот подход для извлечения ключевых слов из текста - KeyBERT.

Все вышеописанные алгоритмы работает довольно хорошо, сравнивать их эффективность не имеет смысла. В качестве стандартного алгоритма для нашего программного комплекса выберем BERT. Аналагично второй подзадаче оставим за пользователем возможность вызова двух других алгоритмов.

Глава 3. Программная реализация

Программный комплекс реализован на высокоуровневом языке программирования общего назначения Python [35]. Был выбран именно этот язык программирования, так как:

- На данный момент Python является одним из самым популярных языков программирования в мире [36]. У него достаточно большое комьюнити, которое постоянно делится своими разработками.
- Для Python существует достаточно много готовых библиотек, реализующих методы обработки естественного языка
  (англ. Natural Language Processing, NLP).
  Библиотеки обработки естественного языка: NLTK, Gensim, CoreNLP, spaCy, TextBlob, PyNLPl.
- Для Python существует достаточно много готовых библиотек, реализующих методы машинного обучения: SciPy, Scikit-learn, Theano , TensorFlow, Keras, PyTorch.

На рисунке 21 представлена схема алгоритма работы программного комплекса. Для решения первой подзадачи по извлечению субтитров из видео используется библиотека YouTubeTranscriptApi. Извлеченные субтитры обрабатываются для дальнейшей работы. Для решения второй подзадачи по сегментации текста на предложения изначально используется библиотек NNsplit, также пользователь может выбрать алгоритмы DeepSegment или punctuator. Для решения третьей подзадачи по сегментации текста с пунктуацией на главы используется алгоритм TextTiling. Для решения четвертой подзадачи по генерации краткого описания глав используется изначально используется KeyBERT, также пользователь может выбрать алгоритмы TextRank и TopicRank.

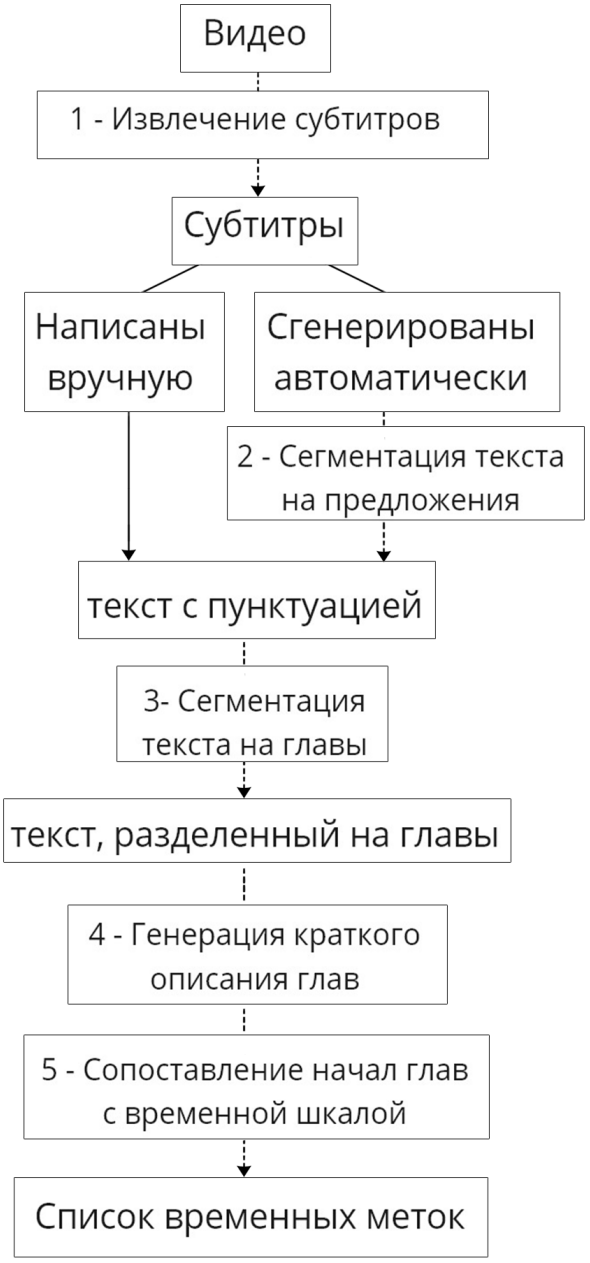

Рис. 21. Схема алгоритма работы программного комплекса.

Опираясь на исходную информации о субтитрах, можно реализовать следующий алгоритм: проверяется вхождение i-ого субтитра в главу, если i+1 субтитр не входит в туже самую главу, то имеет место смена главы. Время начала i+1 субтитра может служить временной меткой начала новой главы.

В итоге программный комплекс, получая на вход ссылку на видео с платформы YouTube, у которого есть субтитры на английском языке, выдает в

качестве выходных данных список, состоящий из временных меток, указывающих на начала глав, и краткого описания глав.

Заключение

В ходе работы были детально рассмотрены методы решения фрагментов поставленной задачи.

Для подзадачи сегментации текста на предложения были рассмотрены существующие методы решения и принцип работы готовых библиотек DeepSegment, NNsplit и Punctuator. Также было произведено сравнение эффективности данных алгоритмов.

Для подзадачи сегментации на главы был произведен анализ существующих подходов, был выбран и рассмотрен алгоритм TextTilling.

Для решения подзадачи генерации краткого описания глав были рассмотрены алгоритмы TextRank, TopicRank, BERT.

Был разработан программный комплекс, решающий задачу автоматической генерации временных меток для видео на платформе YouTube. Ссылка на программный комплекс. [37]

Список литературы

[1] YouTube**.** URL: https://ru.wikipedia.org/wiki/YouTube (дата обращения: 25.05.2021).

[2] tubefilter 05-07-2019. URL: https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/ (дата обращения: 25.05.2021).

[3] YouTube test features and experiments. URL: https://support.google.com/youtube/thread/18138167/youtube-test-features-and-experiments?hl=en (дата обращения: 25.05.2021).

[4] Natural language processing. URL: https://en.wikipedia.org/wiki/Natural_language_processing (дата обращения: 25.05.2021).

[5] YouTube Transcript/Subtitle API. URL: https://github.com/jdepoix/youtube-transcript-api (дата обращения: 25.05.2021).

[6] A. Gravano, M. Jansche, and M. Bacchiani, “Restoring punctuation and capitalization in transcribed speech,” in ICASSP 2009, 2009, pp. 4741–4744.

[7] W. Lu and H. T. Ng, “Better punctuation prediction with dynamic conditional random fields,” in EMNLP 2010, Cambridge, MA, USA, 2010.

[8] N. Ueffing, M. Bisani, and P. Vozila, “Improved models for automatic punctuation prediction for spoken and written text,” in Interspeech 2013, Lyon, France, 2013.

[9] D. Zhang, S. Wu, N. Yang, and M. Li, “Punctuation prediction with transition-based parsing.” in ACL (1), 2013, pp. 752–760.

[10] X. Che, C. Wang, H. Yang, and C. Meinel, “Punctuation prediction for unsegmented transcript based on word vector,” in The 10th International Conference on Language Resources and Evaluation (LREC), 2016.

[11] S. Peitz, M. Freitag, A. Mauser, and H. Ney, “Modeling punctuation prediction as machine translation.” in IWSLT, 2011, pp. 238–245.

[12] E. Cho, J. Niehues, K. Kilgour, and A. Waibel, “Punctuation insertion for real-time spoken language translation,” Proceedings of the Eleventh International Workshop on Spoken Language Translation, 2015.

[13] DeepCorrection 1: Sentence Segmentation of unpunctuated text. URL: https://praneethbedapudi.medium.com/deepcorrection-1-sentence-segmentation-of-unpunctuated-text-a1dbc0db4e98 (дата обращения: 25.05.2021).

DeepSegment. URL: https://github.com/notAI-tech/deepsegment (дата обращения: 25.05.2021).

[14] NNSplit. URL: https://github.com/bminixhofer/nnsplit (дата обращения: 25.05.2021).

[15] Punctuator.// Tilk, O., Alumäe, T. (2016) Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration. Proc. Interspeech 2016, 3047-3051.// URL: https://github.com/ottokart/punctuator2 (дата обращения: 25.05.2021).

[16] Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

[17] Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Comput. 1997. Vol. 9, no. 8. P. 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[18] Schuster, Mike & Paliwal, Kuldip. (1997). Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on. 45. 2673 - 2681. 10.1109/78.650093.

[19] Zhiheng Huang, Wei Xu, & Kai Yu. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging.

[20] Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun & Bengio, Yoshua (2014), Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, arXiv:1412.3555 [cs.NE]

[21] Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano – WildML

[22] SoMaJo/ URL:https://pypi.org/project/SoMaJo/ (дата обращения: 25.05.2021).

[23] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” ICLR2015, arXiv:1409.0473, 2015.

[24] T. Wang and K. Cho, “Larger-context language modelling,” arXiv preprint arXiv:1511.03729, 2015.

[25] European Parliament Proceedings Parallel Corpus 1996-2011. URL: https://www.statmt.org/europarl/ (дата обращения: 25.05.2021).

[26] Pak, Irina & Teh, Phoey. (2018). Text Segmentation Techniques: A Critical Review. 10.1007/978-3-319-66984-7_10.

[27] Koshorek, J. (2018). Text Segmentation as a Supervised Learning Task. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers) (pp. 469–473). Association for Computational Linguistics.

[28] Marti A. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. Comput. Linguist. 23, 1 (March 1997), 33–64.

[29] Extraction-based and Abstraction-based summarization. URL: https://en.wikipedia.org/wiki/Automatic_summarization (дата обращения: 25.05.2021).

[30] Keyword Extraction: from TF-IDF to BERT. URL: https://towardsdatascience.com/keyword-extraction-python-tf-idf-textrank-topicrank-yake-bert-7405d51cd839 (дата обращения: 25.05.2021).

[31] Mihalcea, P. (2004). TextRank: Bringing Order into Text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (pp. 404–411). Association for Computational Linguistics.

[32] PageRank. URL: https://en.wikipedia.org/wiki/PageRank (дата обращения: 25.05.2021).

[33] Bougouin, B. (2013). TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. In Proceedings of the Sixth International Joint Conference on Natural Language Processing (pp. 543–551). Asian Federation of Natural Language Processing.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[35] Python. URL: https://docs.python.org/3.7/ (дата обращения: 25.05.2021).

[36] TIOBE Index for May 2021. URL: https://www.tiobe.com/tiobe-index/(дата обращения: 25.05.2021).

[37] Программный комплекс. URL: https://github.com/Panych-hub/diploma (дата обращения: 25.05.2021).