AI Journey: Школьные тесты

Валентин Малых

SBERBANK

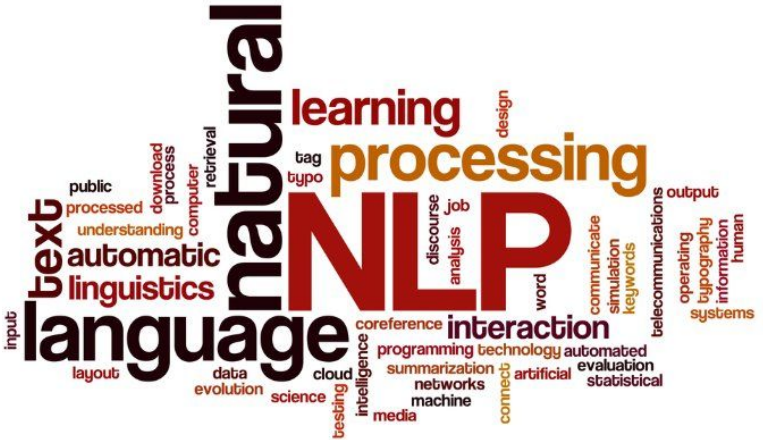
Introduction

Speech & Semantics lab of Noah's Ark Lab dedicates to research and applications of speech and natural language processing, mainly by employing machine learning.

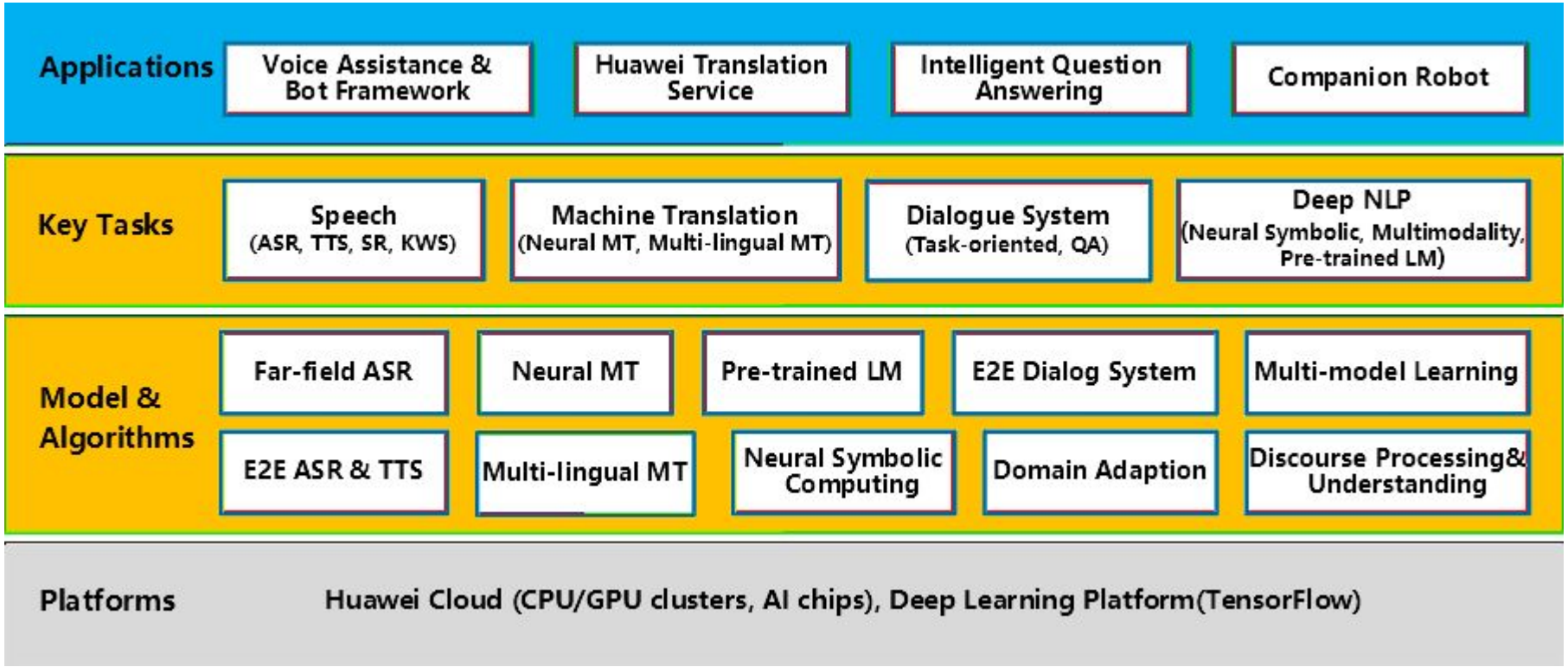
Locations: Hong Kong, Shenzhen, Montreal, Beijing, London, Moscow

Research directions:

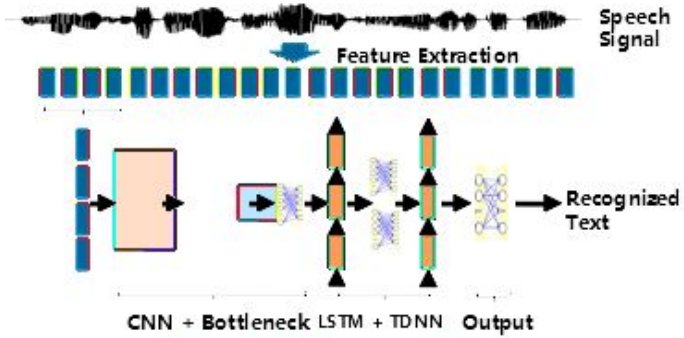
- Speech Recognition & Synthesis
- Machine Translation
- Dialogue System
- Knowledge Graph and Question Answering
- Multi-modal learning
- Pre-trained Language Models
- …



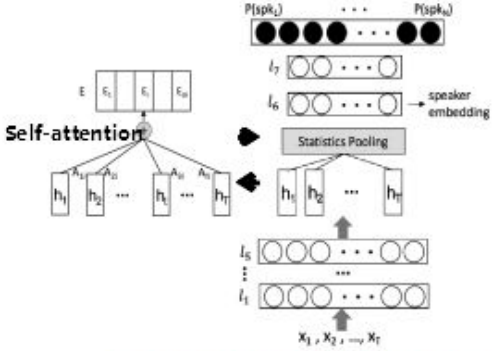
Technical Roadmap



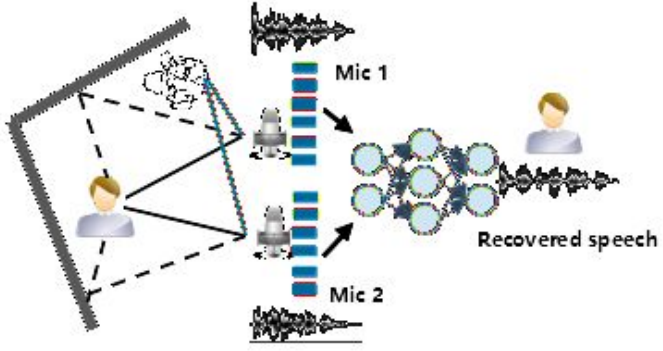
Speech Recognition



Speech Recognition



Speaker Recognition



Speech Enhancement

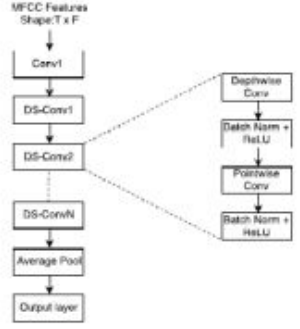
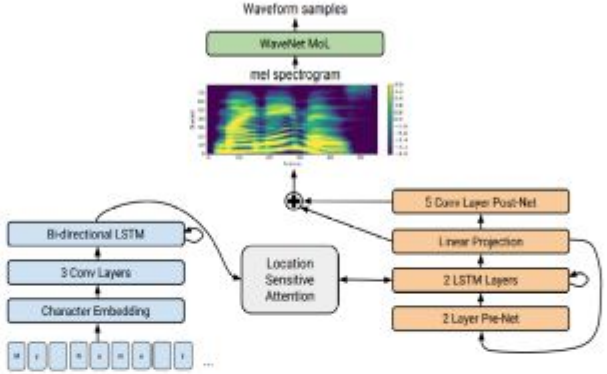


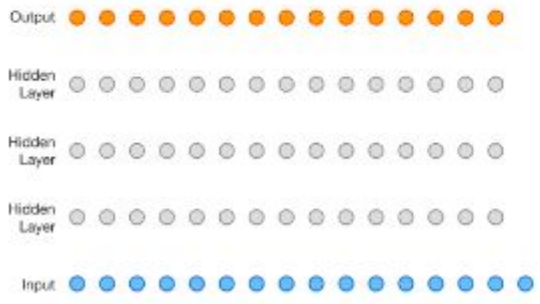
Figure 4: Depthwise separable CNN architecture.

Keyword Spotting

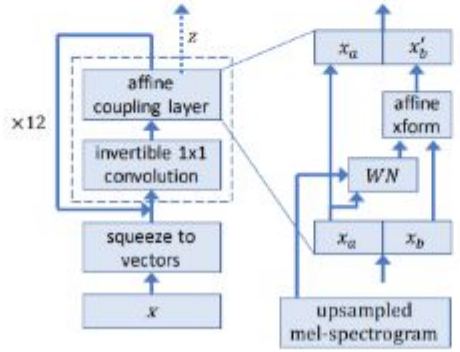
Speech Synthesis

- End-to-end TTS system
- Personalized speech synthesis
- Emotional Speech Synthesis

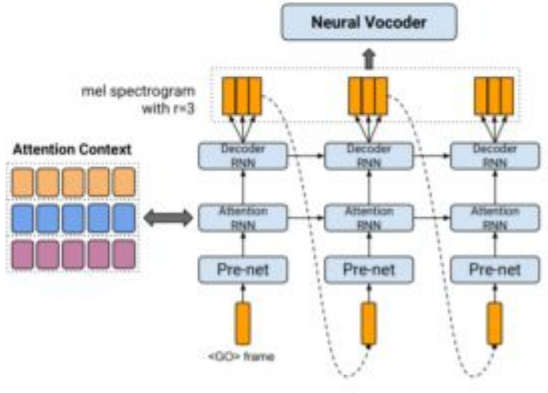




WaveNet



WaveGlow



Emotional TTS

Machine Translation

Transformer

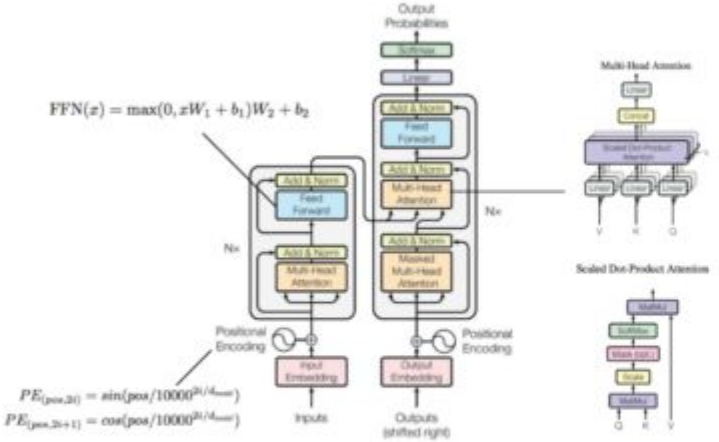
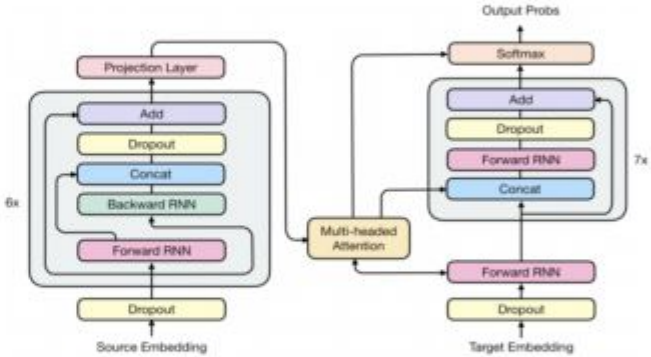


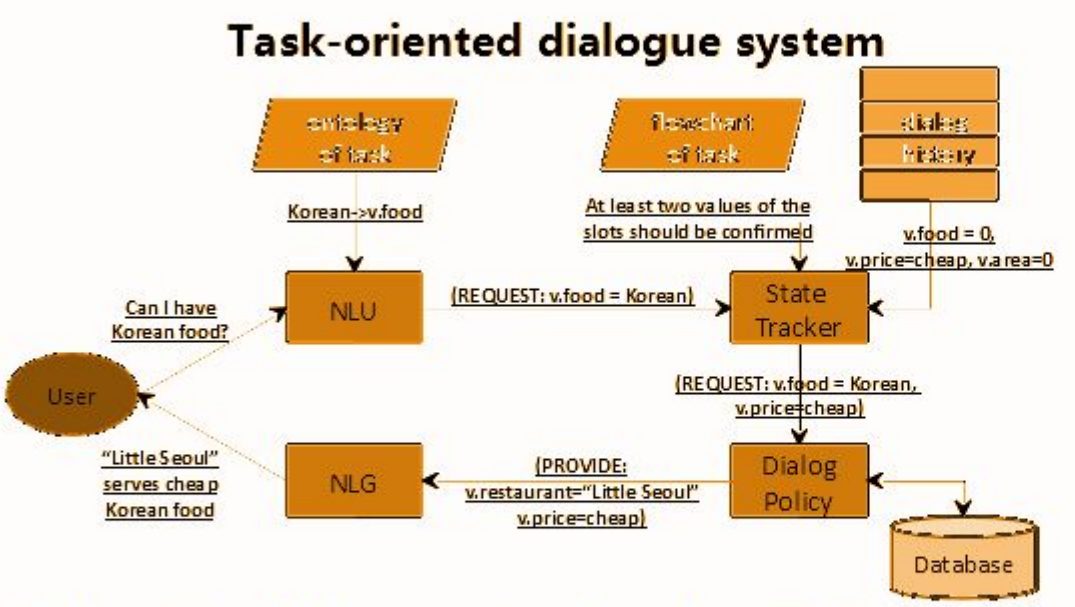
Figure 1: The Transformer - model architecture.

RNMT+



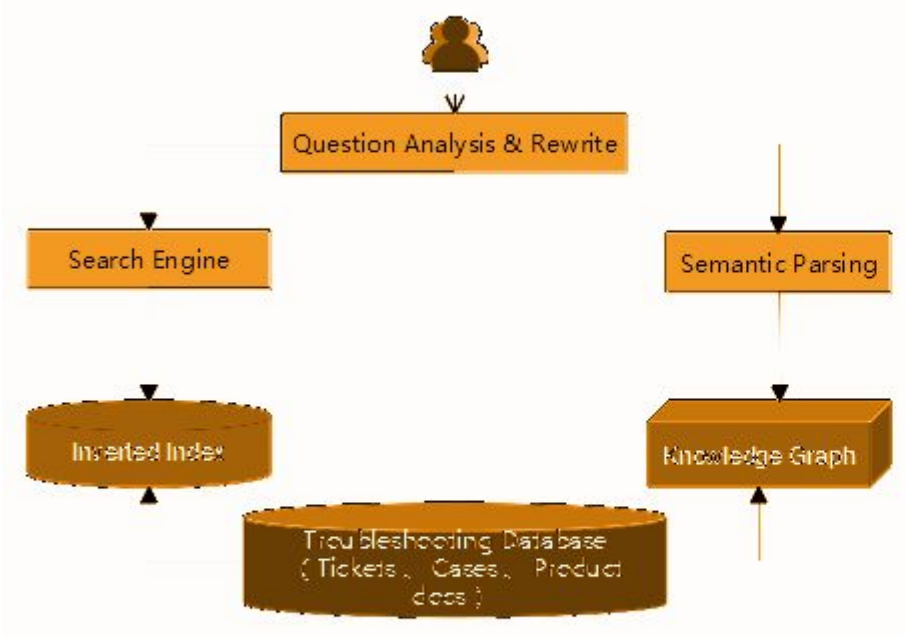
Dialogue System

- Task-oriented dialogue system based on POMDP framework, deployed on Huawei EMUI (Mobile OS)
- Key modules: natural language understanding and generation, dialogue management, context modeling
- Learning methods: supervised learning, self-play, imitation learning and reinforcement learning
- Research topics: multiple intents recognition, semantic parsing, multimodal dialogue systems, hybrid (neural & symbolic) dialogue management



Question Answering

- Retrieval-based QA: based on information retrieval system, enhanced with machine learning techniques, such as learning to rank, deep matching models, etc.
- Knowledge-based QA: extract structured knowledge graph from the product documents, stored in graph database, questions are parsed to database queries through semantic parsing



A Few Papers

1. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In NIPS 2014.
2. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In ACL 2016.
3. Multimodal Convolutional Neural Networks for Matching Image and Sentence. In ICCV 2015.
4. Encoding source language with convolutional neural network for machine translation. In ACL 2015.
5. Neural Responding Machine for Short Text Conversation. In ACL 2015.
6. ERNIE: Enhanced Language Representation with Informative Entities. In ACL 2019.

Призы:

- Команда-победитель получит **1 000 000 ₽**,
  - второе и третье места по **500 000 ₽** и **300 000 ₽** соответственно,
  - четвертое и пятое места получат по **200 000 ₽**,
  - с шестого по десятое место - **100 000 ₽**.

- Кроме того, участникам доступны 2 специальные номинации
  - «Лучшее решение тестовой части» и
  - «Лучшее сочинение»
  - с призами в **150 000 ₽** каждая.

Для нетерпеливых:

Описание соревнования:

- Необходимо разработать алгоритм, который способен успешно **ответить на вопросы экзаменационного теста**, основываясь на информации из открытых источников.
  - Тест состоит из двух частей: непосредственно теста и *сочинения*.
- Участникам предоставляются тестовые варианты заданий, которые можно использовать для валидации решений и для обучения.
- Решения участников отправляются в автоматическую проверяющую систему и оцениваются на скрытом наборе вопросов.

Формат соревнования:

- Решение представляет собой **архив с кодом**, который запускается в контейнерном окружении **Docker**.
- Архивы с решениями загружаются в автоматизированную проверяющую систему. Задания с вопросами проверяются автоматически, задания с написанием сочинений оцениваются профессиональными экспертами раз в неделю.
- **Метрика соревнования** - **суммарный балл** за все задания теста.

Технические ограничения решений:

- Контейнеры изолированы от внешнего мира:
  нет доступа в интернет, нет связи с участниками.
- RAM: **16 Gb**;
- Максимальный размер архива с решением: **20Gb**;
- Максимальный размер Docker-образа (публично доступен): **20Gb**;
- Лимит времени на разогрев (до получения вопросов): **10 минут**
  Это время выделено на подгрузку моделей в оперативную память. После разогрева, решение должно со сравнительно небольшой задержкой отвечать на вопросы.
- Лимит времени на получение ответа:
  - отличается в зависимости от типов вопросов, **≈1 минута / вопрос**.

Процедура проверки решений:

1. Check-фаза

Решение запускается на публично доступном наборе вопросов с известными ответами. Этот запуск необходим для того,чтобы протестировать решение на наличие ошибок в коде и во взаимодействии с проверяющей системой. Вывод **stdout/stderr** и результат проверки полностью доступны участнику.

2. Public Test

Решение запускается на скрытой части вопросов, доступных только организаторам. Порядок вопросов и вариантов ответов в них компонуются случайным образом.

3. Private Test

Решение запускается на финальной скрытой части вопросов, по результатам на которых подводится итог соревнования.

Формат данных:

Экзаменационный билет передается решению в формате JSON.

- text - Текст задания. Возможно использование markdown-style форматирования. Внутри текста могут содержаться ссылки на прикрепленные файлы, например — графические иллюстрации к заданию.
- attachments - Набор прикрепленных файлов (с указанием id, mime-type).
- meta - Мета-информация. Произвольные пары ключ-значение, которые доступны решению и проверочной системе. Предназначено для указания структурированных данные о вопросе. Пример: источник вопроса, предмет экзамена, из которого пришел вопрос.
- answer - Описание формата, в котором необходимо дать ответ. Допускаются разные типы ответов, каждый из которых имеет свои дополнительные параметры и поля:
    - **choice** - выбор одного варианта из списка;
    - **multiple_choice** - выбор подмножества вариантов из списка;
    - **matching** - верное соотнесение объектов из двух множеств;
    - **text** - ответ в виде произвольного текста.
- score - Максимальное количество баллов за задание.

Оценка качества:

Ответ на каждый вопрос оценивается при помощи метрики, соответствующей своему типу вопроса:

- **choice** - accuracy;
- **multiple_choice** - union / intersection;
- **matching** - доля верно сопоставленных вариантов;
- **text** - вызывается специализированная функция оценки качества, либо ответ отправляется на ручную оценку качества.

Итоговый результат формируется путем суммирования баллов за все задания. Затем баллы переводятся в 100-балльную систему по официальной таблице соответствия заданий.

Оценка качества сочинений:

При участии в решении задания написания сочинения, участники получают двухфазную проверку своего решения: сначала автоматическую, затем ручную.

Автоматическая проверка подразумевает поверхностные показатели сгенерированного текста:

- отсутствие плагиата;
- соответствие текста данной теме;
- орфографическая грамотность;
- связность предложений, тавтология;
- наличие речевых ошибок (сленг, мат);
- наличие абзацной структуры;
- выполнение заданного объема (не слишком короткое/длинное).

Автоматическая оценка дается участникам сразу и не является окончательной, а служит для удобства участников.

Ручная проверка осуществляется профессиональными асессорами в соответствии со стандартами.

Базовое решение:

- Классификатор заданий
- Решатели для каждого типа заданий
  - например, 10 задание решается полностью **на правилах**
  - используется **BERT** для embedding
  - **pymorhy2** для морфологического разбора
  - **udpipe** для синтаксического разбора

Пример задания:


```
"id": "10",
     "meta": {
         "language": "ru",
         "source": "            "
     },
     "text": "Укажите варианты ответов, в которых во всех словах одного ряда пропущена одна и та же буква.Запишите номера ответов.",
     "attachments": [],
     "question": {
         "type": "multiple_choice",
         "min_choices": 1,
         "choices": [
             {
                 "id": 1,
                 "text": "пр..града, пр..морье, пр..мудрый"
             },
             {
                 "id": 2,
                 "text": "об..грал, вз..скать, под..тожить"
             },
             {
                 "id": 3,
                 "text": "бе..вкусно, бе..ценный, ра..пустить"
             },
             {
                 "id": 4,
                 "text": "по..скочил, пре..сказание, на..кушенный"
             }
```

Пример решателя:

```
def predict_from_model(self, task):
    result, task = [], standardize_task(task)
    match = re.search(r'буква ([ЭОУАЫЕЁЮЯИ])', task["text"])
    if match:
        letter = match.group(1)
        return self.get_answer_by_vowel(task["question"]["choices"], letter.lower())
    elif "одна и та же буква" in task["text"]:
        for vowel in "эоуаыеёюяидтсэ":
            result_with_this_vowel = self.get_answer_by_vowel(task["question"]["choices"], vowel)
            result.extend(result_with_this_vowel)
    return sorted(list(set(result)))

def get_answer_by_vowel(self, choices, vowel):
    result = list()
    for choice in choices:
        parts = [re.sub(r"^\d\) ?| ?\(.*?\) ?", "", x) for x in choice["parts"]]
        parts = [x.replace("..", vowel) for x in parts]
        if all(self.morph.word_is_known(word) for word in parts): result.append(choice["id"])
    return sorted(result)
```

Решатель для сочинения

- Языковая модель
  - AWD-LSTM на подкорпусе из Тайги
- Тематическое моделирование
  - на корпусе сочинений
- Суммаризация

Решатель для сочинения (2)

- Каждая тема была проинтерпретирована человеком
- Вручную написаны первые фразы
- Подобраны лит. произведения
- Подобраны авторы

Решатель для сочинения (3)

- Сделан шаблон для абзаца
    - первая фраза: 'Автор иллюстрирует данную проблему на примере предложений "{}" и "{}".'
    - вторая фраза: 'На мой взгляд, читатель наблюдает авторскую позицию в предложении: "{}"'

Для терпеливых:

Спасибо за внимание!

@madrugado