cs329s pr1

Problem 0

1. [**1 point**] Spam filtering for emails. FP>FN, bec you may lost an important mail
2. [**1 point**] Spam filtering for search engines. FN>FP. it`s not so bad to lost one of results, but you can keep results clean
3. [**2 points**] Fraud detection for online transactions. FN==FP. In one side you don`t wanna miss a fraud, but in another it`s really frustrating, when your transaction for rent is considered as suspicious.
4. [**1 point**] Classifying skin lesions as benign (negative) or malignant (positive) from a photograph. FN>FP – it may cause harm to the health.

Problem 1: Understanding objectives and models [10 points + 3 points extra credit]

a) speech recognition - siri
machine translation - deepl
predictive texting - autocorrection

b) 1. SR metric – WER. Measure on test set, then on the random set from the real data after deploying. Then we can train classifier for “Tough data” and measure metric on production in two categories: random subset and tough data subset

2. MT – bleu. Measure on test. Add natural labeling in prod (button “Is this translation is correct” and “Propose a translation”). Gather natural labeled sets + get some random prod data and measure metrics on it.

3. PT Autocorrect – gleu (as bleu, but for one lang. 1 seq – sequence with autocorrected words, 2 seq – true sequence). Measure on test set. After deploying – count statistics on the phone, if the user is ok with sending statistics. If the user allows, then gather natural labels.

c) 1. Spotter baseline – CNN (because it will be on edge), ASR – quartznet

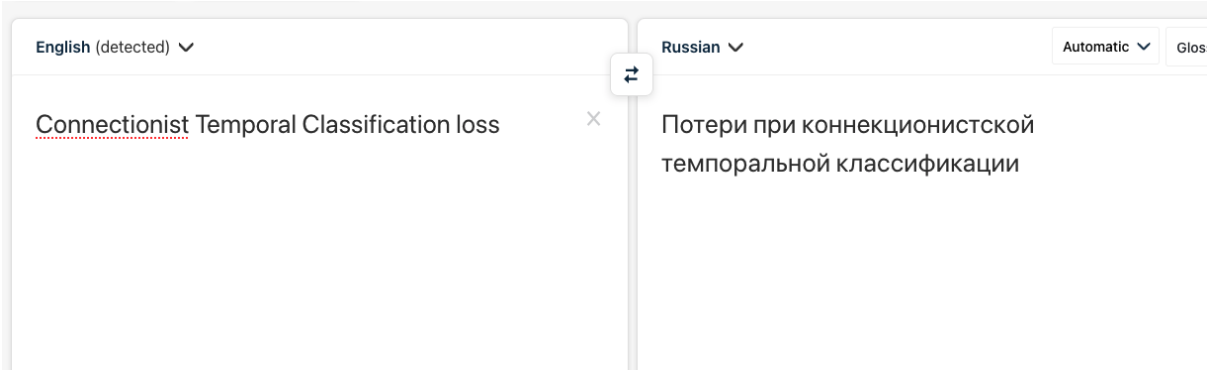
2. Pretrained T5.

3. Dicts for the most frequent words and mapping with common errors with heuristic. If user correct autocorrection -> add correction to banned list.

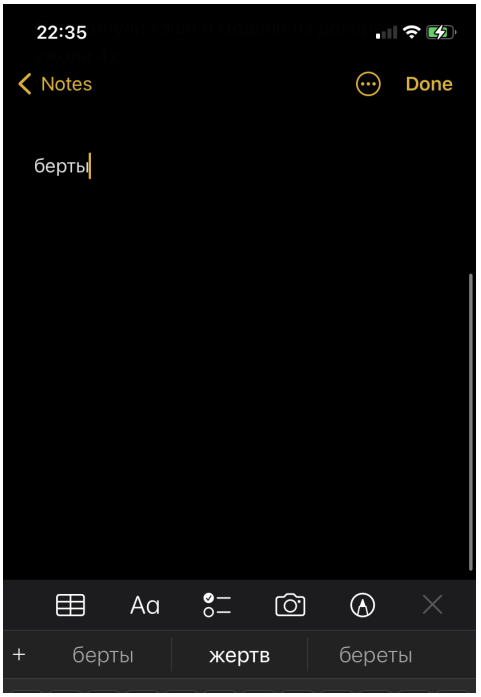
Problem 2: Understanding limitations of ML systems [17 points + 5 points extra credit]

a) 1. Error for accents

2. Error for specific terms.



3. False autocorrect for specific terms.



b) 1. Gather more data with accents, so the model can generalize better. We can use Edge computing not only for spotter, but also for simple cases. If the model is not sure, we can get online predictions from the larger and stronger model from the server.

2. Button for natural labels and correct translation proposals. Gather domain specific data and add to the training set. Add category, so user can set the domain of texts to translate.

3. User specific dicts.

c) System level error. Error for specific terms in machine learning translation sytems.

Problem 3: Bring a research model into production [12 points]

a) https://deepmind.com/research/publications/2022/Red-Teaming-Language-Models-with-Language-Models

The main potentially harmful property of generative language models is that they can generate dangerous text:
+ Toxicity
+ Personal information
+ Discrimination

One of the steps to combat harmful text generation is vulnerability testing. The researchers from DeepMind suggested the novel method, using one more language model, to generate provocative replicas, and a classifier of dangerous replicas, to evaluate target model responses.
Red LM (generates provocative texts) -> Target LM (responds) -> Toxic CLF (evaluates danger of response)

Methods:
1. Zero-shot. (Wrote: "List of questions to ask: 1. ....")
2. Stochastic Few-shot (Took phrases where Target LM screwed up and did a few-shot)
3. Supervised (just picked up the phrases from the previous steps and finetuned Red LM)
4. RL. Build an A2C (actor-critic) system. Actor - LM, critic - Toxic CLF.

As a result, the new LM method was able to find more adversarial phrases.

b) One of the new trends in ML is fairness. The described method will be helpful in testing new language models for products with virtual assistants: **Smart assistants:** Siri, Google Assistant, Alexa, Cortana, etc. and **conversational AI bots:** Mitsuku, Woebot, Replika, etc.

c) **conversational AI bots.**

Fairness is crucial for conversational agents. Our goal is to build robust language models and ensure that the harmful predictions are minimized. For this we will: generate adversarial examples using Red Teaming LM -> test our current Conversational System -> use adversarial examples for fine-tuning more robust Conversational Bot.
Metrics to check. Main metric is N_harmful_responds_to_adverserial/N_adverarial_samples. Also we will check, that new LM will not decrease Conversational metrics:

1) offline: SSA, ACUTE-eval
2) online: messages-per-session, retention, engageness

Considerations:

1) What is harmful in the case of our product? (E.g. If our bot is just a fun service, that copy some famous person, than it's okay, that the Bot will be as toxic, as person.)
2) How robust/cautious our LM should be? (Trade-off: More cautious = more boring)
3) What method of Red LM training to choose? (Start with Supervised on BAD dataset https://aclanthology.org/2021.naacl-main.235.pdf, then RL)

d) MVP: download BAD dataset (https://aclanthology.org/2021.naacl-main.235.pdf), then test the current LM with samples from it. To check the toxicity of the responses of LM we can use a trained classifier, for example, https://huggingface.co/unitary/toxic-bert. For the cases where our model failed - we get the list of words from answers that led our generator to fail. (There are some examples of such words in the article). Then we just ban the outputs containing these words and change the answer with some universal mock. It will be MVP, we don't train anything, and just check and ban problem outputs.

Problem 4: Handling missing values [10 points + 5 points extra credit]

1. Age - missing not at random.

Fare - missing not at random.
df[~df.Fare.isna()].Age.hist()
df[df.Fare.isna()].Age.hist()
Cabin - missing at random. Most of the 2 and 3 class don't have Cabin number.
Embarked - missing completely at random
2.Age - missing not at random. –
Fare - missing not at random. There is free ticket for children < 6 years
Cabin - missing at random, but Most of the 2 and 3 class don't have Cabin number.
3. Age - missing not at random. Median value of the Pclass
Fare - 0, because of free ticket
Cabin - ‘Unknown’ token
Embarked - ‘Unknown’ token