

APPLIED NATURAL LANGUAGE PROCESSING

FINAL PROJECT

Dr. Rachel Shapira

Based on presentations by Amit Mandelbaum

FINAL PROJECT INGREDIENTS

- Project grade will be divided into several components:
 - 10% – Project proposal:
 - 0.5-1 page: Define the project goal, review relevant work, prepare a plan, describe the final outcome.
 - 45% - Project code and results:
 - Quality of code (comments, order, ease of run, clear instructions on how to run)
 - 45% - Project handout:
 - 3-4 pages.
 - Describe the project, data and basic analysis of the data
 - Describe the model you used, the metric chosen and your results
 - Describe additional things that can be done in the future
- Up to 3 students per project
- Proposal submission by 20.2.23
- Submission by 16.3.23

SUGGESTION #1: IMPLEMENT A RESEARCH PAPER

- Implement a research paper on an NLP task
- Most recent research paper are based on transformers which are hard to train so feel free to look into less recent stuff
- Where to look?
 - Google scholar (search for specific tasks and papers that have a lot of citations)
 - NLP benchmarks (see: <https://nlpprogress.com/>)
 - Papers with code (see: <https://paperswithcode.com/>)
- You should achieve results which are at least close to the original paper you are using, so choose wisely.
- Do not use implementation found online (but feel free to look at them for inspirations)
- Summarize the paper and its key novelty as part of your written submission

SUGGESTION #1: IMPLEMENT A RESEARCH PAPER

- Example project:

Implementing the paper: [Generating Gender Augmented Data for NLP](#)

- Write the code, train and run on all datasets mentioned in the paper (and 2 additional ones)
- (optional) Publish the code on github.
- Explain exactly what you did differently...

SUGGESTION #2: USE LEARNT TOOLS ON NEW DATA

- Use the different approaches we learnt on class and find an interesting dataset to use it on
- How to get the data:
 - Collect it your self (preferred): scrape some relevant websites, collect manually, collaborate with someone who owns the data.
 - Get data from the web:
 - Kaggle
 - Papers with code
 - Hugging Face
 - Other places like:
 - <https://machinelearningmastery.com/datasets-natural-language-processing/>
 - <https://github.com/niderhoff/nlp-datasets>

PAPERSWITHCODE DATASETS

835 dataset results for Texts x



Penn Treebank

The English Penn Treebank corpus, and in particular the section of the corpus corresponding to the articles of Wall Street Journal (WSJ), is one of the most known and used corpus for t...
1,545 PAPERS • 10 BENCHMARKS



SQuAD (Stanford Question Answering Dataset)

The Stanford Question Answering Dataset (SQuAD) is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers of questions can be any se...
1,254 PAPERS • 7 BENCHMARKS



Visual Genome

Visual Genome contains Visual Question Answering data in a multi-choice setting. It consists of 101,174 images from MSCOCO with 1.7 million QA pairs, 17 questions per image on aver...
903 PAPERS • 11 BENCHMARKS



GLUE (General Language Understanding Evaluation benchmark)

General Language Understanding Evaluation (GLUE) benchmark is a collection of nine natural language understanding tasks, including single-sentence tasks CoLA and SST-2, similarity...
847 PAPERS • 14 BENCHMARKS



SNLI (Stanford Natural Language Inference)

The SNLI dataset (Stanford Natural Language Inference) consists of 570k sentence-pairs manually labeled as entailment, contradiction, and neutral. Premises are image captions fro...
743 PAPERS • 1 BENCHMARK



CLEVR (Compositional Language and Elementary Visual Reasoning)

CLEVR (Compositional Language and Elementary Visual Reasoning) is a synthetic Visual Question Answering dataset. It contains images of 3D-rendered objects; each image comes...
528 PAPERS • 1 BENCHMARK



Visual Question Answering (VQA)

Visual Question Answering (VQA) is a dataset containing open-ended questions about images. These questions require an understanding of vision, language and commonsense...
435 PAPERS • 2 BENCHMARKS



Billion Word Benchmark

The One Billion Word dataset is a dataset for language modeling. The training/hold-out data was produced from the WMT 2011 News Crawl data using a combination of Bash shell and...
417 PAPERS • 1 BENCHMARK

<https://www.paperswithcode.com/datasets?mod=texts&page=1>

HUGGINGFACE DATASETS

The screenshot displays the Hugging Face Datasets interface. On the left, there are filter sections for Task Category, Task, Language, Multilinguality, Size, and License. The main area on the right shows a list of datasets, with three visible: **acronym_identification**, **ade_corpus_v2**, and **adversarial_qa**. Each dataset entry includes a brief description and a list of metadata tags.

Hugging Face Search models, datasets, users... Models Datasets Pricing Resources Log In Sign Up

Task Category

- conditional-text-generation
- text-classification
- structure-prediction
- sequence-modeling
- question-answering
- text-scoring + 3

Task

- machine-translation
- language-modeling
- named-entity-recognition
- sentiment-classification
- dialogue-modeling
- extractive-qa + 128

Language

- en
- es
- fr
- de
- ru
- ar + 184

Multilinguality

- monolingual
- multilingual
- translation
- other-language-learner

Size

- 10K<n<100K
- 1K<n<10K
- n<1K
- 100K<n<1M
- n>1M
- 1k<10K + 18

License

- mit
- cc-by-4.0
- cc-by-sa-4.0
- cc-by-sa-3.0
- apache-2.0
- cc-by-nc-4.0 + 56

Datasets 638 Search Datasets 11 Sort: Alphabetical

acronym_identification

Acronym identification training and development sets for the acronym identification task at SDU@AAAI-21.

annotations_creators: expert-generated language_creators: found languages: en licenses: mit multilinguality: monolingual size_categories: 10K<n<100K source_datasets: original task_categories: structure-prediction task_ids: structure-prediction-other-acronym-identification

ade_corpus_v2

ADE-Corpus-V2 Dataset: Adverse Drug Reaction Data. This is a dataset for Classification if a sentence is ADE-related (True) or not (False) and Relation Extraction between Adverse Drug Event and Drug. DRUG-AE.rel provides relations between drugs and adverse effects. DRUG-DOSE.rel provides relations between drugs and dosages. ADE-NEG.txt pro...

annotations_creators: expert-generated language_creators: found languages: en licenses: unknown multilinguality: monolingual size_categories: 10K<n<100K size_categories: 1K<n<10K size_categories: n<1K source_datasets: original task_categories: text-classification task_categories: structure-prediction task_categories: structure-prediction task_ids: fact-checking task_ids: coreference-resolution task_ids: coreference-resolution

adversarial_qa

AdversarialQA is a Reading Comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles using an adversarial model-in-the-loop. We use three different models; BiDAF (Seo et al., 2016), BERT-Large (Devlin et al., 2018), and RoBERTa-Large (Liu et al., 2019) in the annotation loop and construct three datasets;...

annotations_creators: crowdsourced language_creators: found languages: en licenses: cc-by-sa-4.0 multilinguality: monolingual size_categories: 10K<n<100K source_datasets: original task_categories: question-answering task_ids: extractive-qa task_ids: open-domain-qa

<https://huggingface.co/datasets>

KAGGLE DATASETS

<https://www.kaggle.com/datasets/tags=13204-NLP>

Datasets


[+ New Dataset](#)[Your Work](#)

[Filters](#)

NLP ×

2,342 Datasets


Hotness ▾



YouTube Videos and Channels Metadata
[The Devastator](#) · Updated 18 days ago
Usability **10.0** · 1 File (CSV) · 86 MB

35


Bronze ...



Netflix TV Shows and Movies (2022 Updated)
[The Devastator](#) · Updated a month ago
Usability **10.0** · 6 Files (CSV) · 2 MB

54


...



FIFA World Cup 2022 Tweets
[Tirendaz Academy](#) · Updated 23 days ago
Usability **10.0** · 1 File (CSV) · 1 MB

34

Bronze ...



Drugs, Side Effects and Medical Condition
[Jithin Varghese](#) · Updated 22 days ago
Usability **9.7** · 1 File (CSV) · 1 MB

27

Bronze ...

SUGGESTION #2: PROJECT EXAMPLE (NEW DATA)

- Text analysis on song lyrics
 - Scraping the web to create a database of 1.2M songs lyrics divided to genres
 - Store in elastic search to be able to search and analyze
 - Do different linguistic analysis (words, complexity etc.)
 - Built a Deep Learning language model that creates lyrics based on genres

SUGGESTION #2: PROJECT EXAMPLE (EXISTING DATA)

- Two interesting datasets:
 - Squad1.1 (Question Answering)
 - <https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/>
 - <https://paperswithcode.com/dataset/squad> (data loaders)
 - WMT (Workshop on Machine Translation)
 - Example (2019): <https://huggingface.co/datasets/wmt19>

SUGGESTION #3: USE SOTA TOOLS TO CREATE INTERESTING APPLICATION

- Use SOTA models (mostly from HuggingFace) to create an app that does something interesting in NLP
- Written material would be on the task, explaining the model and the engineering aspects involved in using this data and bringing up the application
- Nice tutorial (In Hebrew): <https://www.youtube.com/watch?v=MM0HQWx2-iQ&t=1359s>
 - A very recommended channel In general

SUGGESTION #3: SOME EXAMPLES

- Few ideas for projects with GPT3/Transformers:
 - Creating an internal search engine for documents
 - Use sentence embeddings by Transformers and Cosine Similarity
 - Q&A chatbot based on personal summaries.
 - Q&A chatbot based on a podcast (you need to transcribe it first)

TRAINING NEURAL NETWORKS (TIPS FOR #1 AND #2)

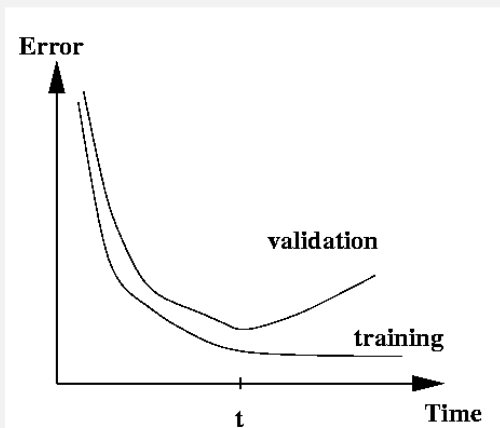
1. Right at the beginning, separate off dev-test and test data splits
2. Define your metric(s):
 - Search online for well established metrics on this task
 - Human evaluation is still much better for summarization
 - You may be able to do at least a very small-scale human eval – ask some friends
3. Establish a baseline
 - Implement the simplest model first (e.g., logistic regression on unigrams and bigrams or averaging word vectors)
 - Compute metrics on train AND dev NOT test
 - Analyze errors
 - If metrics are amazing and no errors:
 - Done! Problem was too easy. Need to restart. 😊/😞

TRAINING NEURAL NETWORKS (TIPS FOR #1 AND #2)

4. Implement existing neural net model
 - Compute metric on train and dev
 - Analyze output and errors
5. Always be close to your data! (Except for the final test set!)
 - Visualize the dataset
 - Collect summary statistics
 - Look at errors
 - Analyze how different hyperparameters affect performance
6. Try different models:
 - Fixed Window FC networks
 - RNNs/LSTMs with or without attention
 - Simple Transformers

OVERFITTING AND FINE-TUNING

- When training, models **overfit** to what you are training on
 - The model correctly describes what happened to occur in particular data you trained on, but the patterns are not general enough patterns to be likely to apply to new data
- The way to monitor and avoid problematic overfitting is using **independent** validation and test sets ...



OVERFITTING AND FINE-TUNING

- You build (estimate/train) a model on a **training set**.
- Often, you then set further hyperparameters on another, independent set of data, the **tuning set**
 - The tuning set is the training set for the hyperparameters!
- You measure progress as you go on a **dev set** (development test set or validation set)
 - If you do that a lot you overfit to the dev set so it can be good to have a second dev set, the **dev2** set
- **Only at the end**, you evaluate and present final numbers on a **test set**
 - Use the final test set **extremely** few times ... ideally only once

OVERFITTING AND FINE-TUNING

- The **train**, **tune**, **dev**, and **test** sets need to be completely distinct
- It is invalid to test on material you have trained on
 - You will get a falsely good performance.
 - We almost always overfit on train
- You need an independent tuning set
 - The hyperparameters won't be set right if tune is same as train
- If you keep running on the same evaluation set, you begin to overfit to that evaluation set
 - Effectively you are “training” on the evaluation set ... you are learning things that do and don't work on that particular eval set and using the info
- To get a valid measure of system performance you need another untrained on, **independent** test set ... hence dev2 and final test

EXPERIMENTAL STRATEGY

- Work incrementally!
- Start with a very simple model and get it to work!
 - It's hard to fix a complex but broken model
- Add bells and whistles one-by-one and get the model working with each of them (or abandon them)
- Initially run on a tiny amount of data
 - You will see bugs much more easily on a tiny dataset
 - Something like 4–8 examples is good
 - Often synthetic data is useful for this
 - Make sure you can get 100% on this data
 - Otherwise, your model is definitely either not powerful enough or it is broken

EXPERIMENTAL STRATEGY

- Train and run your model on a large dataset
 - It should still score close to 100% on the training data after optimization
 - Otherwise, you probably want to consider a more powerful model!
 - Overfitting to training data is **not** something to be scared of when doing deep learning
 - These models are usually good at generalizing because of the way distributed representations share statistical strength regardless of overfitting to training data
- But, still, you now want good generalization performance:
 - Regularize your model until it doesn't overfit on dev data
 - Strategies like L2 regularization can be useful
 - But normally **generous dropout** is the secret to success

RESOURCES

- <https://spacy.io/>
 - Excellent open-source library
 - Tokenization, NER, Sentiment Analysis, Word Vectors, Transformers and much more
- <https://stanfordnlp.github.io/stanza/>
 - Contains all sorts of NLP pipelines (NER, Dependency parsing, Sentiment Analysis and more)
- <https://huggingface.co/>
 - Excellent resources for SOTA models (with easily accessed API)
 - Can sort models by tasks, library and language
 - Sorted by popularity, usage, etc.