State-of-the-Art Transformer Pipelines with spaCy

Daniël de Kok
Biaffine

Madeesh Kannan
Explosion

10th November 2023, aiGrunn

What is spaCy?

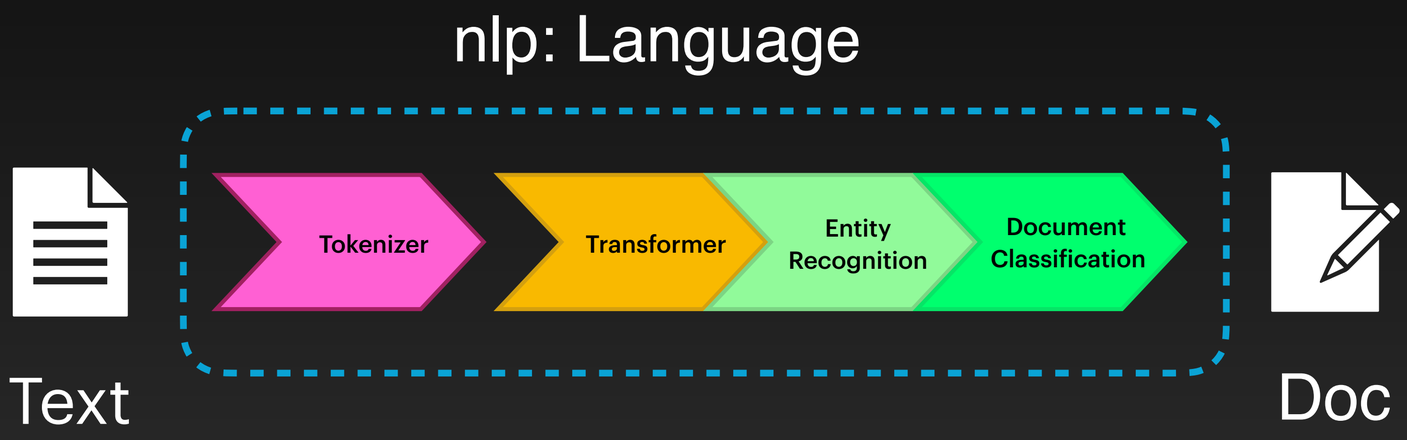
Text documents

Annotated documents

spaCy

```
$ pip install spacy
```

spaCy pipeline



Document container

doc: Doc

On April 25, Twitter's board of directors

unanimously accepted Musk's buyout offer.

Tokenization

doc: Doc

On April 25, Twitter's board of directors unanimously accepted Musk's buyout offer.

doc[9]: Token

class Token:
text: *str*
idx: *int*

Lemmatization

doc: Doc

on director

On April 25, Twitter's board of directors

accept

unanimously accepted Musk's buyout offer.

doc[10].lemma_: str

```
class Token:
    text: str
    lemma_: str
    idx: int
```

Span classification: entity recognition

doc: Doc

DATE ORG

On April 25, Twitter's board of directors

PERSON

unanimously accepted Musk's buyout offer.

doc.ents[2]: Span

```
class Span:
    text: str
    label_: str
    start: int
    end: int
    start_char: int
    end_char: int
```

Span classification: entity recognition

doc.ents[2]: Span

int
int

Document classification

doc: Doc

On April 25, Twitter's board of directors

unanimously accepted Musk's buyout offer.

Newswire

doc.cats: Dict[str, float]

How transformers are used

Hidden Representations

Station to Station is the tenth studio album by David Bowie .

SpanCat

Lemmatizer

How transformers are used

Hidden Representations

Station to Station is the tenth studio album by David Bowie .

SpanCat

Lemmatizer

Softmax

How transformers are used

Hidden Representations

Station to Station is the tenth studio album by David Bowie .

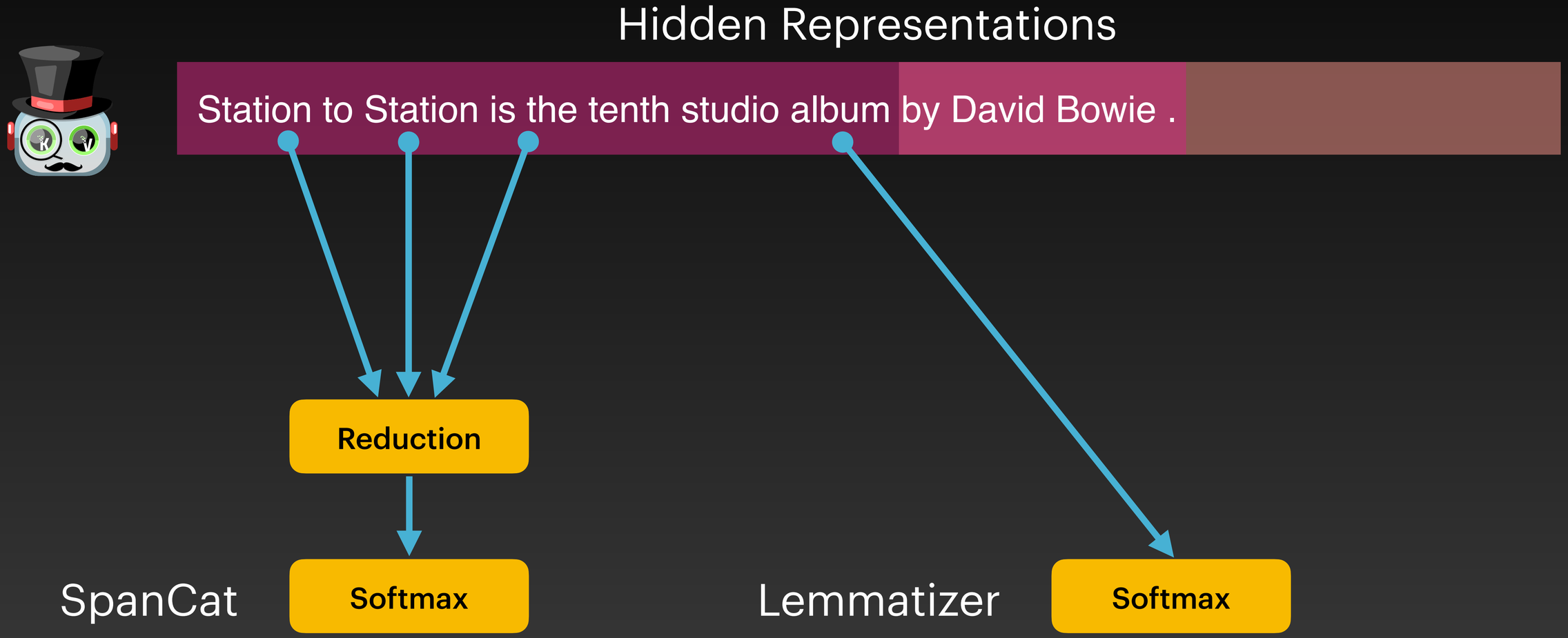
Reduction

SpanCat

Lemmatizer

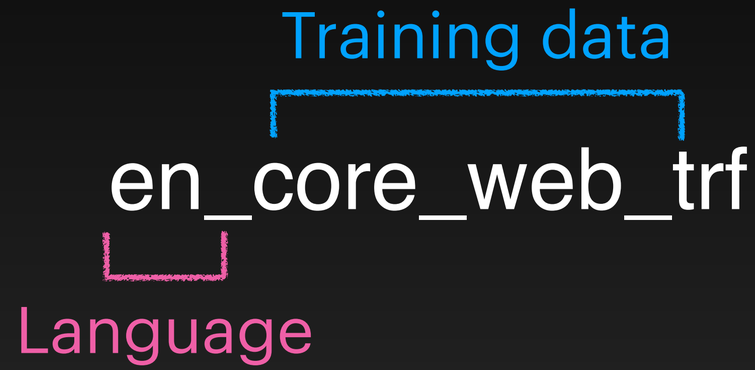
Softmax

How transformers are used



Inference with Transformer Pipelines

Pretrained transformer models




```
$ spacy download en_core_web_trf
```

```
nlp = spacy.load("en_core_web_trf")
```

Small webservice

- Small FastAPI REST web service with /annotate endpoint.
- The user provides texts in JSON format.
- Process the texts with the en_core_web_trf pipeline.
- Return:
  - Tokens
  - Lemmas
  - Entities

Endpoint

```
en_core_web_trf = spacy.load("en_core_web_trf")

@app.post("/annotate")
def annotate(docs: List[str]):
    annotated_docs = en_core_web_trf.pipe(docs)
    return [convert_spacy_doc(doc) for doc in annotated_docs]
```

Data model

```
@dataclass
class Token:
    text: str
    lemma: str

@dataclass
class Span:
    text: str
    label: str
    span: Tuple[int, int]
```

```
@dataclass
class Annotations:
    tokens: List[Token]
    entities: List[Span]
```

Data conversion

```
def convert_spacy_doc(doc: Doc):
    tokens = [Token(text=token.text, lemma=token.lemma_)
              for token in doc]
    entities = [
        Span(text=ent.text,
             label=ent.label_,
             span=(ent.start, ent.end)
        )
        for ent in doc.ents
    ]
    return Annotations(tokens=tokens, entities=entities)
```

Example run

```
% curl \
  -X POST \
  -H "Content-Type: application/json" \
  -d '["The Glove80 was released by MoErgo in 2023, bringing joy to
many typists."]' \
  http://localhost:8000/annotate | jq '.[].entities'
```


```
[
  {
    "text": "Glove80",
    "label": "PRODUCT",
    "span": [1, 2]
  },
  {
    "text": "MoErgo",
    "label": "ORG",
    "span": [5, 6]
  },
  {
    "text": "2023",
    "label": "DATE",
    "span": [7, 8]
  }
]
```

Training Transformer Pipelines

Training data

- Groningen Meaning Bank (GMB):
  - Developed by the University of Groningen.
  - >10K annotated English texts.
    - Genres: Predominantly newspaper text.
    - Public domain.
  - Annotations: Named Entities, Lemmas, Part-of-Speech

Pipeline configuration

- Full description of the pipeline.
- Structured, pluggable sections.
- Reproducibility.
- Validation.

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

Pipeline configuration

Sections

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

Pipeline configuration

Sections

```ini
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

`[paths]` - Paths to data and other assets.

Pipeline configuration

Sections

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

`[system]` - Settings related to system and hardware.

Pipeline configuration

Sections

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

`[nlp]` - Pipeline components, language and tokenizer.

Pipeline configuration

Sections

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

`[components]` - Definitions of pipeline components.

Pipeline configuration

Sections

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

`[corpora]` - Readers for the training data.

Pipeline configuration

Sections

```
[paths]
train = null
dev = null

[system]
gpu_allocator = "pytorch"
seed = 1

[nlp]
lang = "en"
pipeline = ["transformer","tagger", "lemmatizer", "ner"]
disabled = []
batch_size = 64
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.lemmatizer]
factory = "trainable_lemmatizer"
min_tree_freq = 3
overwrite = false
scorer = {"@scorers":"spacy.lemmatizer_scorer.v1"}
top_k = 1

[corpora]

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
gold_preproc = false
max_length = 0
limit = 0

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system:seed}
gpu_allocator = ${system:gpu_allocator}
dropout = 0.1
accumulate_gradient = 3
patience = 5000
max_epochs = 5
max_steps = 10000
eval_frequency = 250
```

`[training]` - Training parameters.

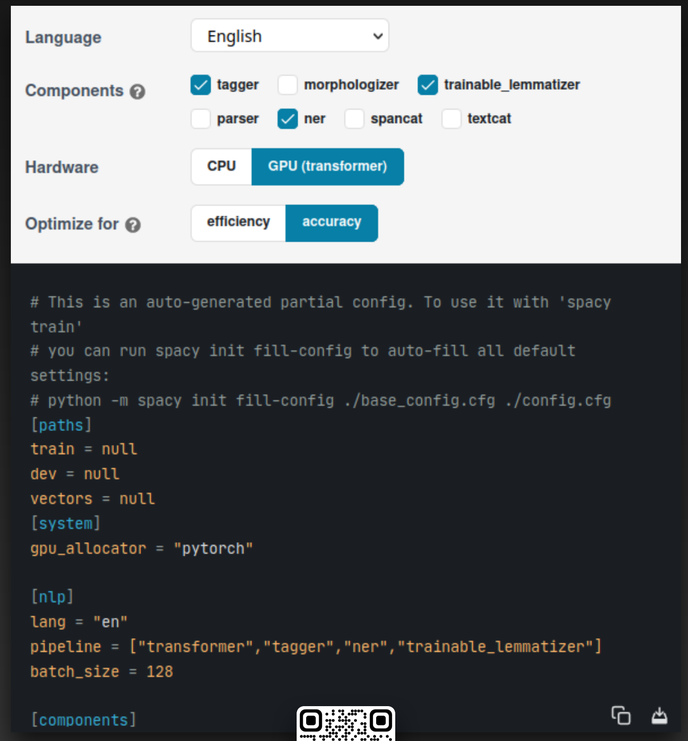
Pipeline configuration

Command-line:

```
$ spacy init config base.cfg
```

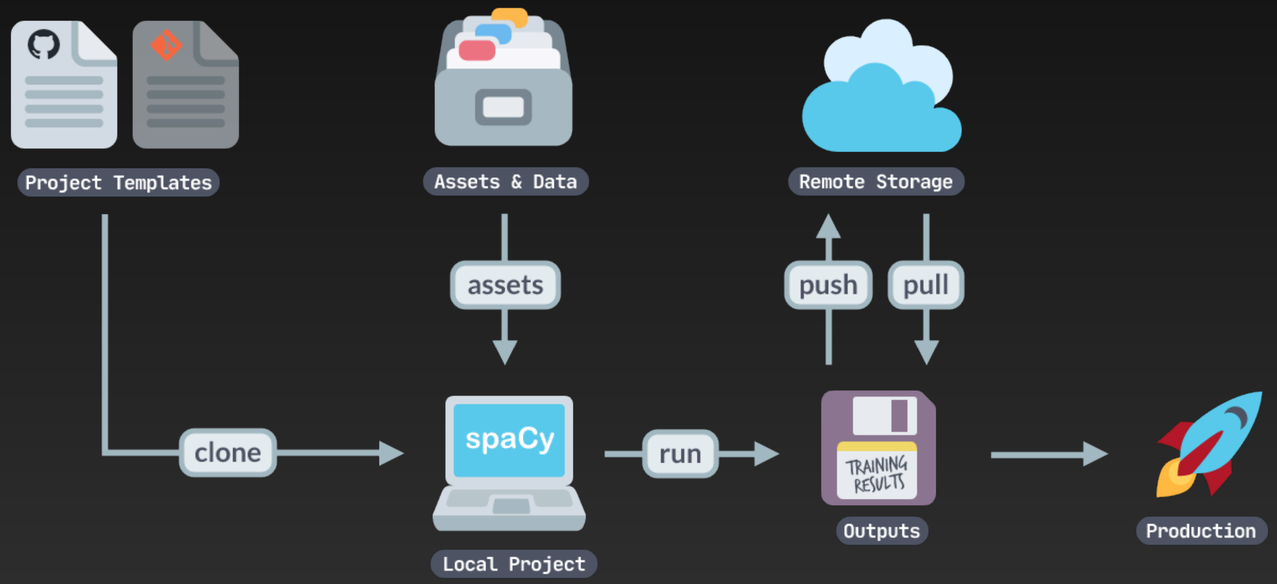
```
$ spacy init fill-config base.cfg
filled.cfg
```

Quickstart widget on spacy.io



Projects

- End-to-end spaCy workflows.
  - Assets.
  - Training.
  - Evaluation.
  - Deployment.



Projects
Definition

```yaml
title: "spaCy Transformer POS/Lemmatizer/NER pipeline"

# Variables can be referenced across the project.yml using ${vars.var_name}
vars:
  seed: 1
  splits: "80|10|10"


directories: ["assets", "training", "configs", "metrics", "corpus"]


assets:
  - dest: "assets/gmb.zip"
    url: "https://gmb.let.rug.nl/releases/gmb-2.2.0.zip"
    checksum: "7b12fd710826d5a4963e57a43100f6c6"

commands:
  - name: corpus
    help: "Convert the data to spaCy's format"
    script:
      - 'unzip assets/gmb.zip "*.tags" -d assets/'
      - >-
        python scripts/conv.py assets/gmb-2.2.0 corpus
        --seed ${vars.seed}
        --splits '${vars.splits}'
    deps:
      - "assets/gmb.zip"
    outputs:
      - "corpus/train.spacy"
      - "corpus/dev.spacy"
      - "corpus/test.spacy"

workflows:
  all:
    - corpus
    - train
    - evaluate
```

Projects
Definition

```yaml
title: "spaCy Transformer POS/Lemmatizer/NER pipeline"

# Variables can be referenced across the project.yml using ${vars.var_name}
vars:
  seed: 1
  splits: "80|10|10"


directories: ["assets", "training", "configs", "metrics", "corpus"]


assets:
  - dest: "assets/gmb.zip"
    url: "https://gmb.let.rug.nl/releases/gmb-2.2.0.zip"
    checksum: "7b12fd710826d5a4963e57a43100f6c6"

commands:
  - name: corpus
    help: "Convert the data to spaCy's format"
    script:
      - 'unzip assets/gmb.zip "*.tags" -d assets/'
      - >-
        python scripts/conv.py assets/gmb-2.2.0 corpus
        --seed ${vars.seed}
        --splits '${vars.splits}'
    deps:
      - "assets/gmb.zip"
    outputs:
      - "corpus/train.spacy"
      - "corpus/dev.spacy"
      - "corpus/test.spacy"

workflows:
  all:
    - corpus
    - train
    - evaluate
```

`vars` - Configurable settings.

Projects
Definition

```yaml
title: "spaCy Transformer POS/Lemmatizer/NER pipeline"

# Variables can be referenced across the project.yml using ${vars.var_name}
vars:
  seed: 1
  splits: "80|10|10"


directories: ["assets", "training", "configs", "metrics", "corpus"]


assets:
  - dest: "assets/gmb.zip"
    url: "https://gmb.let.rug.nl/releases/gmb-2.2.0.zip"
    checksum: "7b12fd710826d5a4963e57a43100f6c6"

commands:
  - name: corpus
    help: "Convert the data to spaCy's format"
    script:
      - 'unzip assets/gmb.zip "*.tags" -d assets/'
      - >-
        python scripts/conv.py assets/gmb-2.2.0 corpus
        --seed ${vars.seed}
        --splits '${vars.splits}'
    deps:
      - "assets/gmb.zip"
    outputs:
      - "corpus/train.spacy"
      - "corpus/dev.spacy"
      - "corpus/test.spacy"

workflows:
  all:
    - corpus
    - train
    - evaluate
```

`directories` - Directories for outputs, assets, etc.

Projects
Definition

```yaml
title: "spaCy Transformer POS/Lemmatizer/NER pipeline"

# Variables can be referenced across the project.yml using ${vars.var_name}
vars:
  seed: 1
  splits: "80|10|10"


directories: ["assets", "training", "configs", "metrics", "corpus"]


assets:
  - dest: "assets/gmb.zip"
    url: "https://gmb.let.rug.nl/releases/gmb-2.2.0.zip"
    checksum: "7b12fd710826d5a4963e57a43100f6c6"

commands:
  - name: corpus
    help: "Convert the data to spaCy's format"
    script:
      - 'unzip assets/gmb.zip "*.tags" -d assets/'
      - >-
        python scripts/conv.py assets/gmb-2.2.0 corpus
        --seed ${vars.seed}
        --splits '${vars.splits}'
    deps:
      - "assets/gmb.zip"
    outputs:
      - "corpus/train.spacy"
      - "corpus/dev.spacy"
      - "corpus/test.spacy"

workflows:
  all:
    - corpus
    - train
    - evaluate
```

`assets` - List of asset definitions.

Projects
Definition

```yaml
title: "spaCy Transformer POS/Lemmatizer/NER pipeline"

# Variables can be referenced across the project.yml using ${vars.var_name}
vars:
  seed: 1
  splits: "80|10|10"


directories: ["assets", "training", "configs", "metrics", "corpus"]


assets:
  - dest: "assets/gmb.zip"
    url: "https://gmb.let.rug.nl/releases/gmb-2.2.0.zip"
    checksum: "7b12fd710826d5a4963e57a43100f6c6"

commands:
  - name: corpus
    help: "Convert the data to spaCy's format"
    script:
      - 'unzip assets/gmb.zip "*.tags" -d assets/'
      - >-
        python scripts/conv.py assets/gmb-2.2.0 corpus
        --seed ${vars.seed}
        --splits '${vars.splits}'
    deps:
      - "assets/gmb.zip"
    outputs:
      - "corpus/train.spacy"
      - "corpus/dev.spacy"
      - "corpus/test.spacy"

workflows:
  all:
    - corpus
    - train
    - evaluate
```

`commands` - List of named commands, their dependencies and outputs.

Projects
Definition

```yaml
title: "spaCy Transformer POS/Lemmatizer/NER pipeline"

# Variables can be referenced across the project.yml using ${vars.var_name}
vars:
  seed: 1
  splits: "80|10|10"


directories: ["assets", "training", "configs", "metrics", "corpus"]


assets:
  - dest: "assets/gmb.zip"
    url: "https://gmb.let.rug.nl/releases/gmb-2.2.0.zip"
    checksum: "7b12fd710826d5a4963e57a43100f6c6"

commands:
  - name: corpus
    help: "Convert the data to spaCy's format"
    script:
      - 'unzip assets/gmb.zip "*.tags" -d assets/'
      - >-
        python scripts/conv.py assets/gmb-2.2.0 corpus
        --seed ${vars.seed}
        --splits '${vars.splits}'
    deps:
      - "assets/gmb.zip"
    outputs:
      - "corpus/train.spacy"
      - "corpus/dev.spacy"
      - "corpus/test.spacy"

workflows:
  all:
    - corpus
    - train
    - evaluate
```

`workflows` - Lists of commands to execute in-order.

Let's train a spaCy pipeline with:

Lemmatiser, Named Entity Recogniser and Part-of-Speech Tagger

Training

Project

Structure:

Commands:

- 📚 `corpus`
- 💪 `train`
- ✅ `evaluate`

Training

Step 1: Download assets

Training

Step 1: Download assets

```
$ cd project
$ spacy project assets

ℹ Fetching 1 asset(s)
✔ Downloaded asset
/Users/mkannan/dev/aiGrunn-2023/project/assets/gmb.zip
```

```
$ cd project
$ spacy project assets
```

Training

Step 2: Convert corpus into training data

Training

Step 2: Convert corpus into training data

```
$ spacy project run corpus

=================================== corpus ===================================
Running command: unzip assets/gmb.zip '*.tags' -d assets/
Archive:  assets/gmb.zip
  inflating: assets/gmb-2.2.0/data/p76/d0310/en.tags
  inflating: assets/gmb-2.2.0/data/p76/d0689/en.tags
  inflating: assets/gmb-2.2.0/data/p76/d0687/en.tags
  ...
Running command: /Users/mkannan/opt/miniconda3/bin/python scripts/gmb_converter.py assets/gmb-2.2.0 corpus
--seed 1 --splits '80|10|10' --n-docs 5000

Input directory: assets/gmb-2.2.0
Output directory: corpus
Splits: (80, 10, 10)
Number of documents to convert: 5000
Coalesce consecutive named entities: True
Processing documents...
100%|████████████████████████████████████████| 10000/10000 [00:08<00:00, 1186.21it/s]
Converting 5000 documents...
Wrote 4000 documents to the training set
Wrote 500 documents to the development set
Wrote 500 documents to the test set
```

$ spacy project run corpus

Training

Step 3: Train the pipeline

Training

Step 3: Train the pipeline

```
$ spacy project run train

=================================== train ===================================
Running command: /Users/mkannan/opt/miniconda3/bin/python -m spacy train configs/default.cfg -o training/ --
gpu-id 0 --paths.train corpus/train.spacy --paths.dev corpus/dev.spacy
ℹ Saving to output directory: training
ℹ Using GPU: 0

============================ Initializing pipeline =============================
✔ Initialized pipeline

============================== Training pipeline ===============================
ℹ Pipeline: ['transformer', 'tagger', 'lemmatizer', 'ner']
ℹ Initial learn rate: 0.0
E    #       LOSS TRANS...  LOSS TAGGER  LOSS LEMMA     LOSS NER  TAG_ACC  LEMMA_ACC  ENTS_F  SPEED   SCORE
---  ------  -------------  -----------  -------------  --------  -------  ---------  ------  ------  ------
0       0       11019.05       700.33         715.70    516.79     0.10      64.94    0.63   35326.51    0.30
0     250      541568.05    409296.48      403073.66  101223.78    54.00      66.10   76.57   38424.30    0.70
1     500       35135.77     99152.09      115594.67   22673.97    97.38      89.71   80.39   36970.75    0.86
2     750       23021.88     14806.34       37780.36   19576.18    98.38      96.88   83.10   35357.75    0.91
3    1000       39303.60     10942.97       15616.03   17299.31    98.42      98.20   83.85   41808.22    0.92
4    1250       31570.73      9091.26        9232.81   16120.65    98.51      98.64   84.20   41526.73    0.92
Epoch 5:  96%|                                                    | 241/250 [01:45<00:04,  2.21it/s]
✔ Saved pipeline to output directory
training/model-last
```

$ spacy project run train

Training

Step 4: Evaluate on held-out data

Training

Step 4: Evaluate on held-out data

```
$ spacy project run evaluate

=================================== evaluate ===================================
Running command: /Users/mkannan/opt/miniconda3/bin/python -m spacy evaluate
./training/model-best ./corpus/test.spacy --output ./metrics/default.json --gpu-id 0
ℹ Using GPU: 0

=================================== Results ====================================
TOK     -
TAG     98.39
LEMMA   98.66
NER P   84.69
NER R   84.32
NER F   84.50
SPEED   21740
============================== NER (per type) ==============================
              P       R       F
GEO-NAM    87.16   87.51   87.34
TIM-DOW    98.78  100.00   99.39
PER-NAM    71.52   78.80   74.98
TIM-DAT    75.17   65.22   69.84
TIM-MOY    99.01  100.00   99.50
GPE-NAM    96.38   92.65   94.48
TIM-CLO    90.00   45.00   60.00
ART-NAM     0.00    0.00    0.00
ORG-NAM    67.94   70.99   69.43
PER-FAM    87.16   86.96   87.06
PER-GIV    87.07   73.72   79.84

✔ Saved results to metrics/default.json
```

$ spacy project run evaluate

Using the trained model

```python
import spacy

# Load the pipeline we just trained.
nlp = spacy.load("training/model-last")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. “I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn’t "
        "worth talking to,” said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp.pipe([text])

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

💫 spaCy - spacy.io

💥 Explosion - explosion.ai

🪁 Biaffine - biaffine.ai

🔗 Daniël - danieldk.eu

in Madeesh - @m-kannan

Play around with the project!

github.com/explosion/aiGrunn-2023

Curated Transformers

💫 spaCy - spacy.io

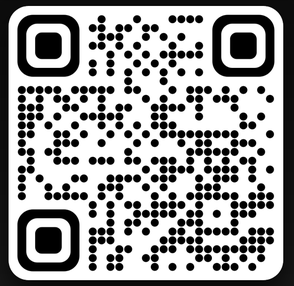
💥 Explosion - explosion.ai

🪂 Biaffine - biaffine.ai

🔗 Daniël - danieldk.eu

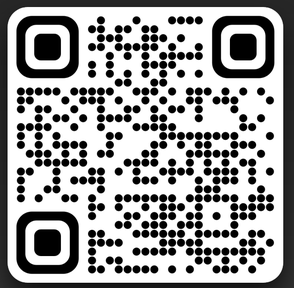
in Madeesh - @m-kannan

Play around with the project!



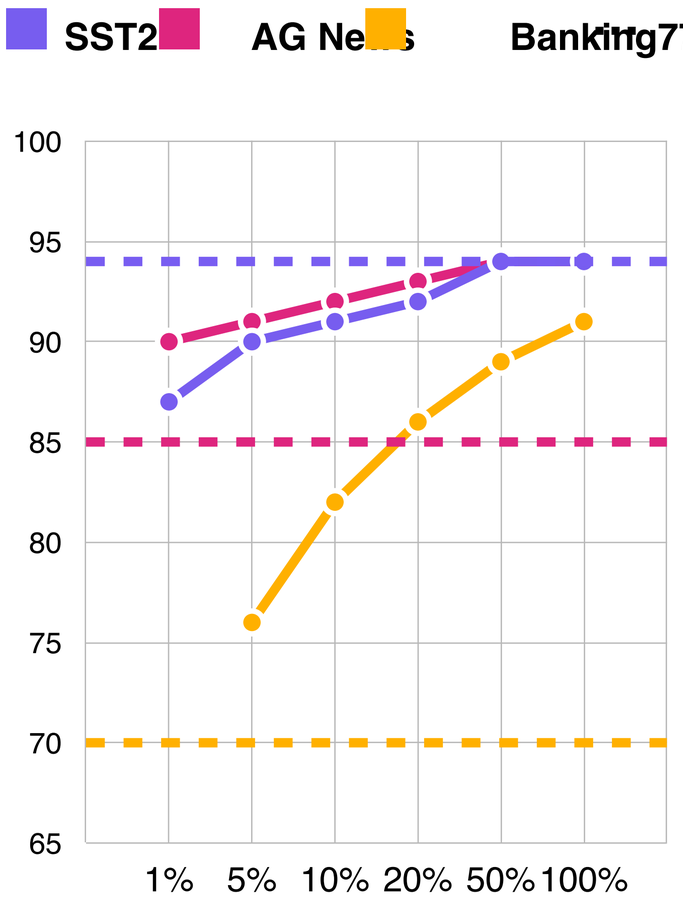
github.com/explosion/aiGrunn-2023

Curated Transformers 🤖

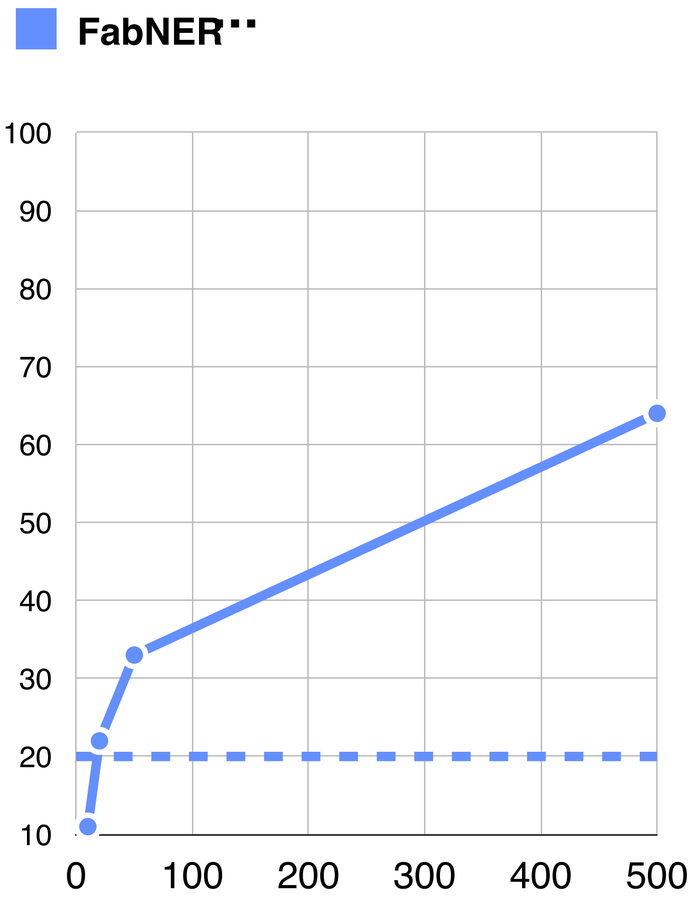


```
$ pip install curated-transformers
```

Text Classification

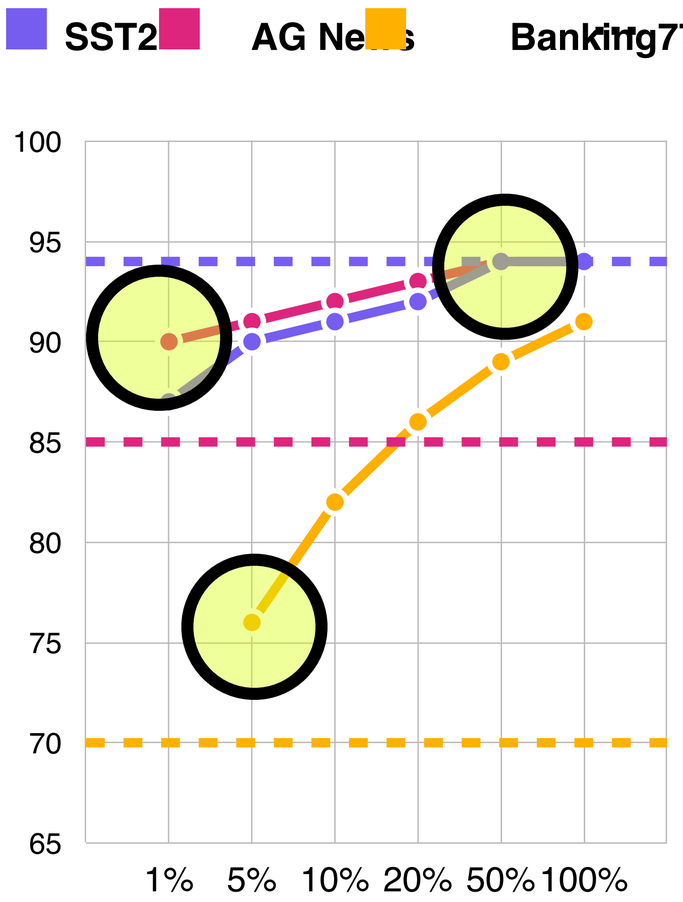


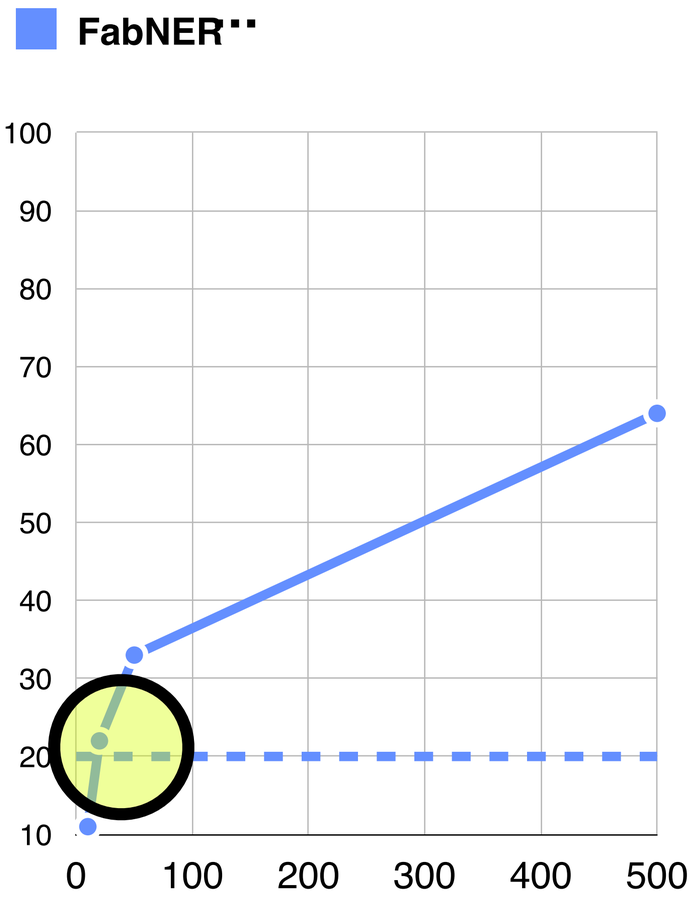
Entity Recognition



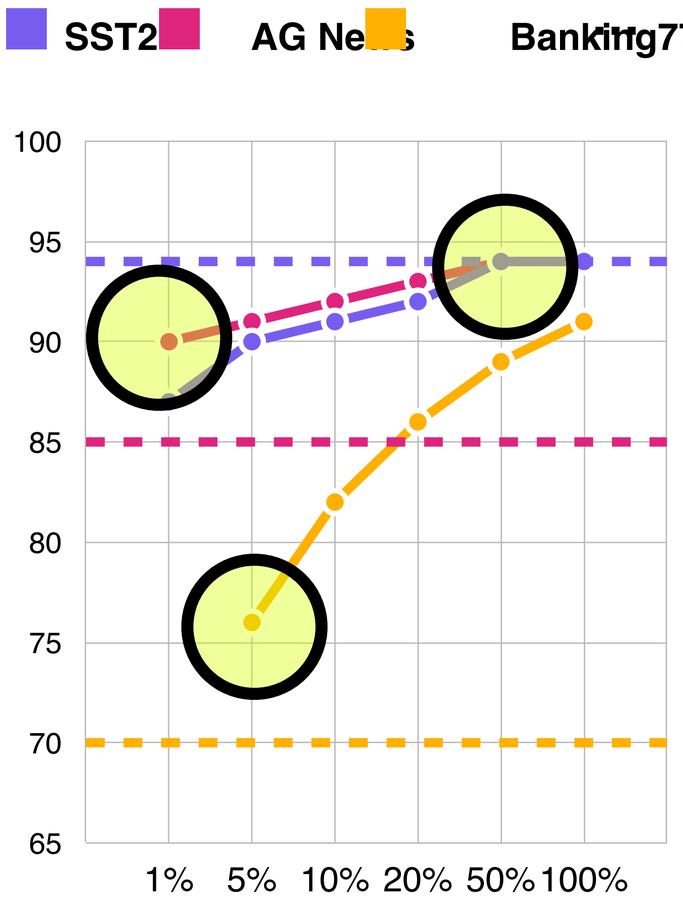
Text Classification

Entity Recognition





Text Classification



Entity Recognition