https://campus.datacamp.com/courses/advanced-nlp-with-spacy/training-a-neural-network-model?ex=1

04.01

Training and updating models

ADVANCED NLP WITH SPACY

Ines Montani
spaCy core developer

Why updating the model?

- Better results on your specific domain --> You can usually make the model more accurate by showing it examples from your domain.
- Learn classification schemes specifically for your problem -->You often also want to predict categories specific to your problem, so the model needs to learn about them.
- Essential for text classification
- Very useful for named entity recognition
- Less critical for part-of-speech tagging and dependency parsing

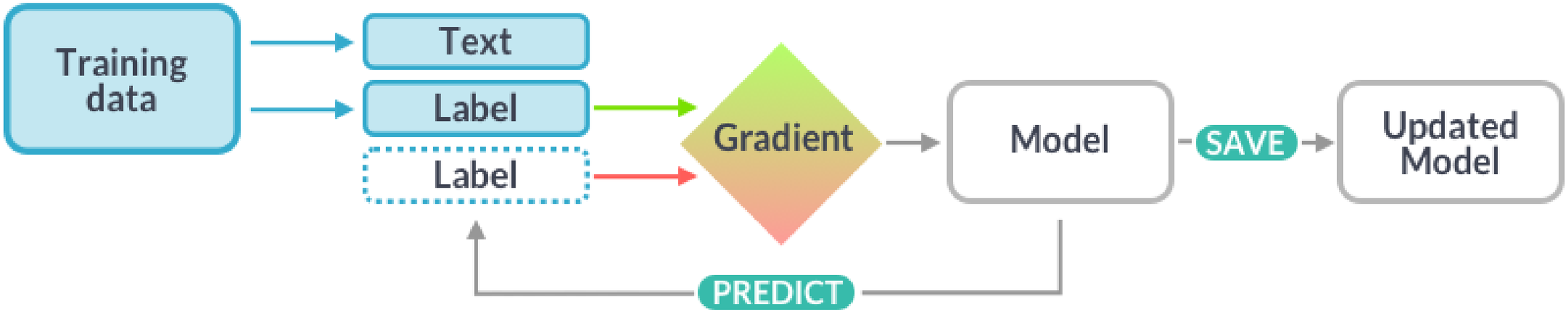
How training works (1)

spaCy supports updating existing models with more examples, and training new models.

If we are NOT starting with a pre-trained model, we first initialize the weights randomly.

1. **Initialize** the model weights randomly with `nlp.begin_training`
2. **Predict** a few examples with the current weights by calling `nlp.update`
3. **Compare** prediction with true labels
4. **Calculate** how to change weights to improve predictions
5. **Update** weights slightly
6. Go back to 2.

How training works (2)



- **Training data:** Examples and their annotations.
- **Text:** The input text the model should predict a label for.
- **Label:** The label the model should predict.
- **Gradient:** How to change the weights.

Example: Training the entity recognizer

- The entity recognizer tags words and phrases in context
- Each token can only be part of one entity
- Examples need to come with context

```
("iPhone X is coming", {'entities': [(0, 8, 'GADGET')]})
```

  - Texts with no entities are also important

```
("I need a new phone! Any tips?", {'entities': []})
```

  - **Goal:** teach the model to generalize

The training data

- Examples of what we want the model to predict in context
- Update an **existing model**: a few hundred to a few thousand examples
- Train a **new category**: a few thousand to a million examples
  - spaCy's English models: 2 million words
- Usually created manually by human annotators
- Can be semi-automated – for example, using spaCy's `Matcher` !

To update an existing model, we can start with a few hundred to a few thousand examples.

To train a new category we may need up to a million.

Let's practice!

ADVANCED NLP WITH SPACY

https://campus.datacamp.com/courses/advanced-nlp-with-spacy/training-a-neural-network-model?ex=5

04.05

The training loop

ADVANCED NLP WITH SPACY

Ines Montani
spaCy core developer

The steps of a training loop

1. **Loop** for a number of times.
2. **Shuffle** the training data.
3. **Divide** the data into batches.
4. **Update** the model for each batch.
5. **Save** the updated model.

While some other libraries give you one method that takes care of training a model, spaCy gives you full control over the training loop.

The training loop is a series of steps that's performed to train or update a model.

(1) We usually need to perform it several times, for multiple iterations, so that the model can learn from it effectively.
If we want to train for 10 iterations, we need to loop 10 times.

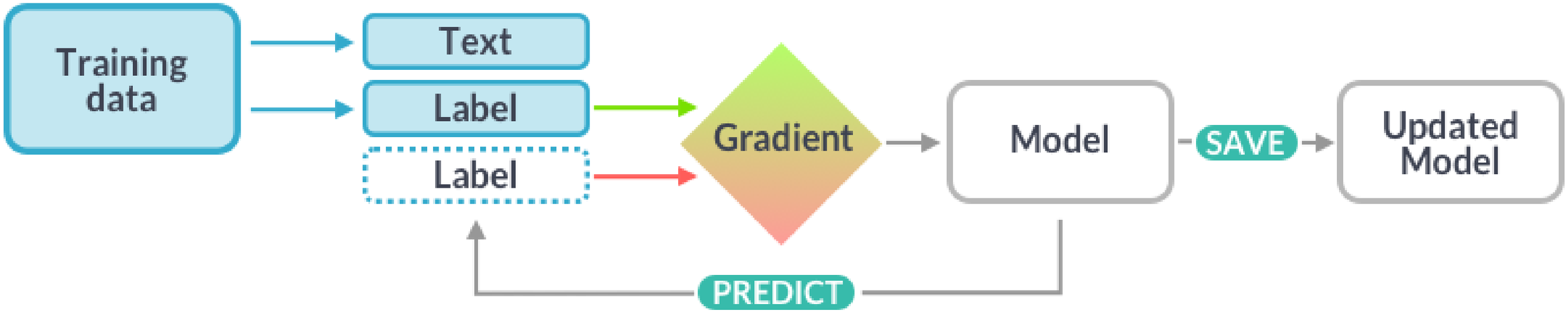
(2) To prevent the model grom getting stuck in a suboptimal solution, we ramdomly shuffle the data for each iteration. This is a very common strategy when doing stochastic gradient descent.

(3) Next, we divide the training data into batches of several examples, also known as minibatching. This makes it easier to make a more accurate estimate of the gradient.

(4) Finally, we update the model for each batch, and start the loop again until we've reached the last iteration.

(5) We can then save the model to a directory and use it in spaCy.

Recap: How training works



- **Training data:** Examples and their annotations. --> They are examples we want to update the model with.
- **Text:** The input text the model should predict a label for. --> The text should be a sentence, paragraph or longer document. For the best result, it should be similar to what the model will see at runtime.
- **Label:** The label the model should predict. --> This can be a text category, or an entity span andd its type.
- **Gradient:** How to change the weights. --> The gradient is how we should change the model to reduce the current error. It's computed when we compare the predicted label to the true label.

Example loop

Let's imagine we have a list of training examples consisting of texts and entity annotations.

```
TRAINING_DATA = [
    ("How to preorder the iPhone X", {'entities': [(20, 28, 'GADGET')]})
    # And many more examples...
]
```

```
# Loop for 10 iterations
for i in range(10):
    # Shuffle the training data
    random.shuffle(TRAINING_DATA)
    # Create batches and iterate over them
    for batch in spacy.util.minibatch(TRAINING_DATA):
        # Split the batch in texts and annotations
        texts = [text for text, annotation in batch]
        annotations = [annotation for text, annotation in batch]
        # Update the model
        nlp.update(texts, annotations)
# Save the model
nlp.to_disk(path_to_model)
```

Updating an existing model

- Improve the predictions on new data
- Especially useful to improve existing categories, like `PERSON`
- Also possible to add new categories
- Be careful and make sure the model doesn't "forget" the old ones

spaCy lets you update an existing pre-trained model with more date, for example, to improve its predictions on different texts.

This is especially useful if you waant to improve categories the model already knows, like "person" or "organization".

You can also update a model to add new categories. Just make sure to always update it with example of the new category and example of the other categories it previously predicted correctly.

Otherwise improving the new category might hurt the other categories.

Setting up a new pipeline from scratch

```
# Start with blank English model
nlp = spacy.blank('en')
# Create blank entity recognizer and add it to the pipeline
ner = nlp.create_pipe('ner')
nlp.add_pipe(ner)
# Add a new label
ner.add_label('GADGET')
# Start the training
nlp.begin_training()
# Train for 10 iterations
for itn in range(10):
    random.shuffle(examples)
    # Divide examples into batches
    for batch in spacy.util.minibatch(examples, size=2):
        texts = [text for text, annotation in batch]
        annotations = [annotation for text, annotation in batch]
        # Update the model
        nlp.update(texts, annotations)
```

In this example, we start off with a blank English model using the spacy.blank method.

The blank model doesn't have any pipeline components, only the language data and tokenization rules.

Let's practice!

ADVANCED NLP WITH SPACY

https://campus.datacamp.com/courses/advanced-nlp-with-spacy/training-a-neural-network-model?ex=9

04.09

Best practices for training spaCy models

ADVANCED NLP WITH SPACY

Ines Montani
spaCy core developer

Problem 1: Models can "forget" things

- Existing model can overfit on new data
  - e.g.: if you only update it with `WEBSITE`, it can "unlearn" what a `PERSON` is
- Also known as "catastrophic forgetting" problem

Solution 1: Mix in previously correct predictions

- For example, if you're training `WEBSITE`, also include examples of `PERSON`
- Run existing spaCy model over data and extract all other relevant entities

BAD:

```
TRAINING_DATA = [
    ('Reddit is a website', {'entities': [(0, 6, 'WEBSITE')]})
]
```

GOOD:

```
TRAINING_DATA = [
    ('Reddit is a website', {'entities': [(0, 6, 'WEBSITE')]}),
    ('Obama is a person', {'entities': [(0, 5, 'PERSON')]})
]
```

Problem 2: Models can't learn everything

- spaCy's models make predictions based on **local context**
- Model can struggle to learn if decision is difficult to make based on context
- Label scheme needs to be consistent and not too specific
  - For example: `CLOTHING` is better than `ADULT_CLOTHING` and `CHILDRENS_CLOTHING`

Solution 2: Plan your label scheme carefully

- Pick categories that are reflected in local context
- More generic is better than too specific
- Use rules to go from generic labels to specific categories

BAD:

```
LABELS = ['ADULT_SHOES', 'CHILDRENS_SHOES', 'BANDS_I_LIKE']
```

GOOD:

```
LABELS = ['CLOTHING', 'BAND']
```

Let's practice!

ADVANCED NLP WITH SPACY

https://campus.datacamp.com/courses/advanced-nlp-with-spacy/training-a-neural-network-model?ex=12

04.12

Wrapping up

ADVANCED NLP WITH SPACY

Ines Montani
spaCy core developer

Your new spaCy skills

- Extract **linguistic features**: part-of-speech tags, dependencies, named entities
- Work with pre-trained **statistical models**
- Find words and phrases using `Matcher` and `PhraseMatcher` **match rules**

→ chapter 1

- Best practices for working with **data structures** `Doc` , `Token` `Span` , `Vocab` , `Lexeme`
- Find **semantic similarities** using **word vectors**

→ chapter 2

- Write custom **pipeline components** with **extension attributes**
- **Scale up** your spaCy pipelines and make them fast

→ chapter 3

- Create **training data** for spaCy' statistical models
- **Train and update** spaCy's neural network models with new data

→ chapter 4

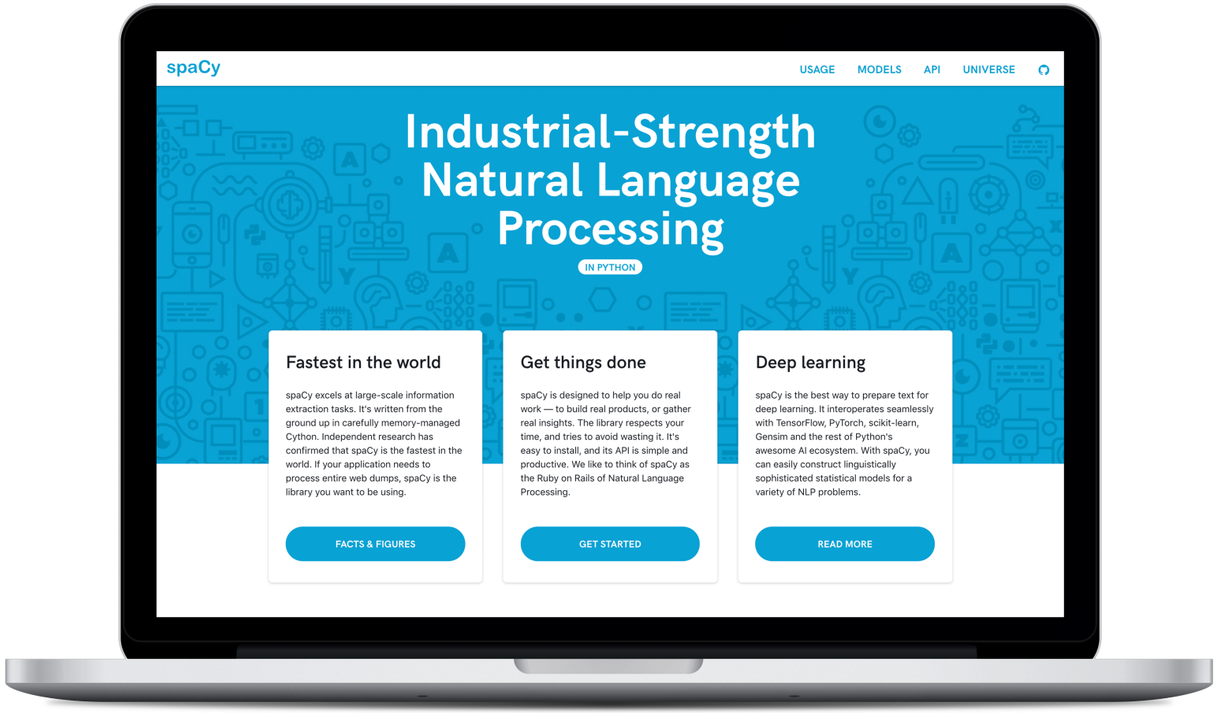
More things to do with spaCy (1)

- **Training and updating** other pipeline components
  - Part-of-speech tagger
  - Dependency parser
  - Text classifier

More things to do with spaCy (2)

- **Customizing the tokenizer**
  - Adding rules and exceptions to split text differently
- **Adding or improving support for other languages**
  - 45+ languages currently
  - Lots of room for improvement and more languages
  - Allows training models for other languages

See the website for more info and documentation!



spacy.io

Thanks and see you soon!

ADVANCED NLP WITH SPACY