A Practical Introduction to Machine Learning in Python
Day 5 – Friday
»Next steps«

Damian Trilling
Anne Kroon

d.c.trilling@uva.nl, @damian0604
a.c.kroon@uva.nl, @annekroon

September 30, 2021

Gesis

This part: State of the art and next steps

- Hot and happening: Transformers
  - Do I need all this fancy stuff?
- Your takeaway

Hot and happening: Transformers

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- (Huge) pre-trained model (by, e.g., Google) that is fine-tuned for specific task (by you)
- In simple neural networks, identical words have identical meanings – but meaning can depend on context
- Therefore, the model should take context into accounts. For example, in LSRTM we use *sequences* of words.
- But meaning of a word does not necessarily depend *sequentially* on the preceeding words in that order
- Solution: *Learn* which tokens *to attend to* (attention)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- (Huge) pre-trained model (by, e.g., Google) that is fine-tuned for specific task (by you)
- In simple neural networks, identical words have identical meanings – but meaning can depend on context
- Therefore, the model should take context into accounts. For example, in LSRTM we use *sequences* of words.
- But meaning of a word does not necessarily depend *sequentially* on the preceeding words in that order
- Solution: *Learn* which tokens *to attend to* (attention)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- (Huge) pre-trained model (by, e.g., Google) that is fine-tuned for specific task (by you)
- In simple neural networks, identical words have identical meanings – but meaning can depend on context
- Therefore, the model should take context into accounts. For example, in LSRTM we use *sequences* of words.
- But meaning of a word does not necessarily depend *sequentially* on the preceeding words in that order
- Solution: *Learn* which tokens *to attend to* (attention)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- (Huge) pre-trained model (by, e.g., Google) that is fine-tuned for specific task (by you)
- In simple neural networks, identical words have identical meanings – but meaning can depend on context
- Therefore, the model should take context into accounts. For example, in LSRTM we use *sequences* of words.
- But meaning of a word does not necessarily depend *sequentially* on the preceeding words in that order
- Solution: *Learn* which tokens *to attend to* (attention)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- (Huge) pre-trained model (by, e.g., Google) that is fine-tuned for specific task (by you)
- In simple neural networks, identical words have identical meanings – but meaning can depend on context
- Therefore, the model should take context into accounts. For example, in LSRTM we use *sequences* of words.
- But meaning of a word does not necessarily depend *sequentially* on the preceeding words in that order
- Solution: *Learn* which tokens *to attend to* (attention)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- We can use BERT for a lot of different tasks: for sequence-to-sequence predictions (e.g., translation), but also for classification (yeah!)
- Can be done in keras
- The Huggingface transformers library includes a lot of BERT-models (e.g., BERTje for Dutch)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- We can use BERT for a lot of different tasks: for sequence-to-sequence predictions (e.g., translation), but also for classification (yeah!)
- Can be done in keras
- The Huggingface transformers library includes a lot of BERT-models (e.g., BERTje for Dutch)

The idea

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- We can use BERT for a lot of different tasks: for sequence-to-sequence predictions (e.g., translation), but also for classification (yeah!)
- Can be done in keras
- The Huggingface transformers library includes a lot of BERT-models (e.g., BERTje for Dutch)

Let's look at an example (imdb.ipynb)

Hot and happening: Transformers

Do I need all this fancy stuff?

Things to consider

How important is. . .

- precision/recall? Am I satisfied with .88 when .90 is theoretically possible? .85? .80? .75?
- explainability?
- computational ressources?
- generalizability and out-of-sample performance?

Things to consider

How important is. . .

- precision/recall? Am I satisfied with .88 when .90 is theoretically possible? .85? .80? .75?
- explainability?
- computational ressources?
- generalizability and out-of-sample performance?

Things to consider

How important is. . .

- precision/recall? Am I satisfied with .88 when .90 is theoretically possible? .85? .80? .75?
- explainability?
- computational ressources?
- generalizability and out-of-sample performance?

Things to consider

How important is. . .

- precision/recall? Am I satisfied with .88 when .90 is theoretically possible? .85? .80? .75?
- explainability?
- computational ressources?
- generalizability and out-of-sample performance?

Do I need all this fancy stuff?

- Always estimate a simple baseline model first
- Invest in good hyperparameter-tuning (cross-validation, gridsearch) and don't forget to set aside unseen data for the *final* evaluation.
- If you (a) need to get the highest possible accuracy, or (b) have reasons to assume that the model does not generalize well enough (overfitting problems, bad out-of-sample prediction (e.g., training topics on newspaper 1, predicting topics in newspaper 2)), try embedding-based approaches, transformers, etc.
- Rule of thumb: the more abstract/latent what you want to predict, the less likely classic ML is going to work

Do I need all this fancy stuff?

- Always estimate a simple baseline model first
- Invest in good hyperparameter-tuning (cross-validation, gridsearch) and don't forget to set aside unseen data for the *final* evaluation.
- If you (a) need to get the highest possible accuracy, or (b) have reasons to assume that the model does not generalize well enough (overfitting problems, bad out-of-sample prediction (e.g., training topics on newspaper 1, predicting topics in newspaper 2)), try embedding-based approaches, transformers, etc.
- Rule of thumb: the more abstract/latent what you want to predict, the less likely classic ML is going to work

Do I need all this fancy stuff?

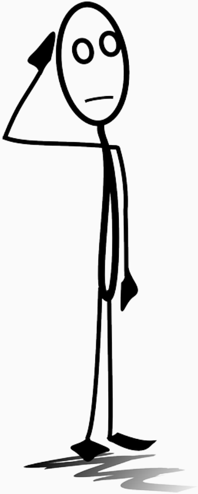
- Always estimate a simple baseline model first
- Invest in good hyperparameter-tuning (cross-validation, gridsearch) and don't forget to set aside unseen data for the *final* evaluation.
- If you (a) need to get the highest possible accuracy, or (b) have reasons to assume that the model does not generalize well enough (overfitting problems, bad out-of-sample prediction (e.g., training topics on newspaper 1, predicting topics in newspaper 2)), try embedding-based approaches, transformers, etc.
- Rule of thumb: the more abstract/latent what you want to predict, the less likely classic ML is going to work

Do I need all this fancy stuff?

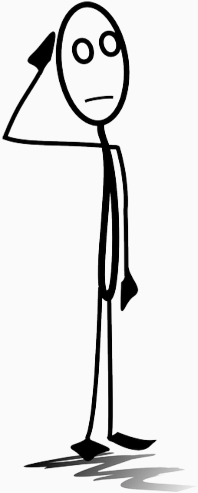
- Always estimate a simple baseline model first
- Invest in good hyperparameter-tuning (cross-validation, gridsearch) and don't forget to set aside unseen data for the *final* evaluation.
- If you (a) need to get the highest possible accuracy, or (b) have reasons to assume that the model does not generalize well enough (overfitting problems, bad out-of-sample prediction (e.g., training topics on newspaper 1, predicting topics in newspaper 2)), try embedding-based approaches, transformers, etc.
- Rule of thumb: the more abstract/latent what you want to predict, the less likely classic ML is going to work

Your takeaway

(short recap of course)



Have your plans about how to and wether to use ML changed?



What are your next steps?

Last part: we help you working on (or discussing about) your own projects.

References

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Naacl hlt 2019 - 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies - proceedings of the conference*.