Pytech Summit

All about Python 2021

Szymon Moliński

Prosty model == prosta implementacja.
Nie w TensorFlow.

SALA B

Godz. 16:30

idego intel lingaro NOKIA Unit8 DC DEVELOPER COUNCIL

Revolut TOOPLOOX STXNEXT python powerhouse accenture DataArt

SalesIntelligence°
DIGITREE GROUP

Prosty model == Prosta implementacja. Nie w TensorFlow!

Szymon Moliński
Data Scientist
szymon.molinski@salesintelligence.pl
Digitree Group S.A

Research project **E-commerce Shopping Patterns Prediction System** was funded under Priority Axis 1.1 of Smart Growth Operational Programme 2014-2020 co-funded by European Regional Development Fund. Project number: **POIR.01.01.01-00-0632/18**

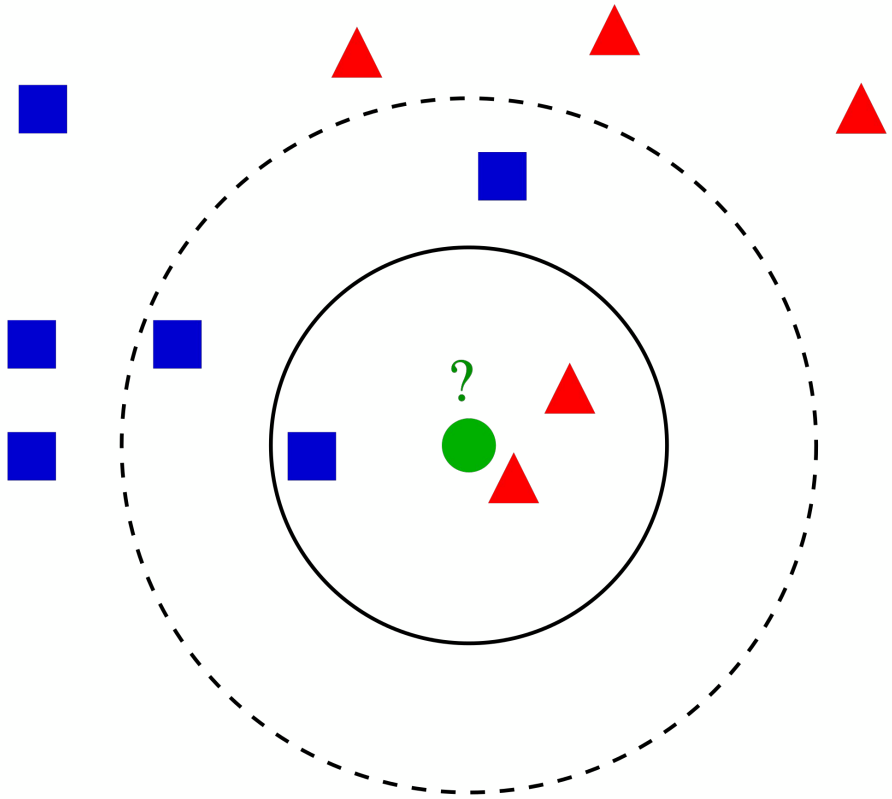
European Funds
Smart Growth

Republic of Poland

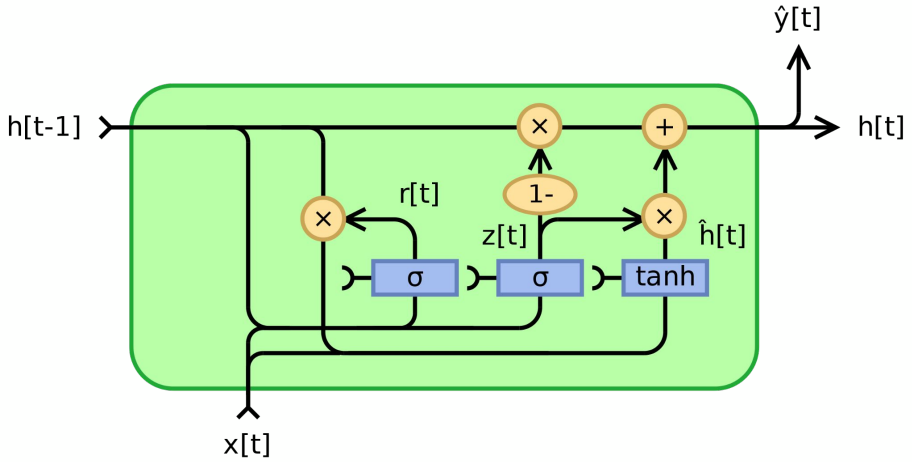
European Union
European Regional
Development Fund

Czym jest *Prosty Model*?

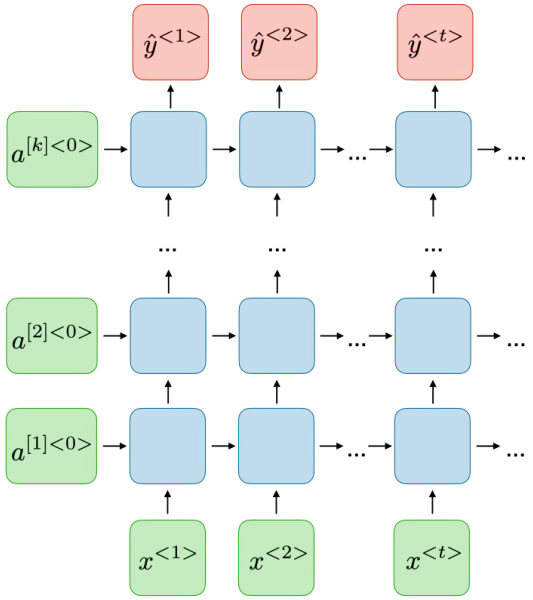
Definicja "prostoty"



By Antti Ajanki AnAj - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2170282



By Jeblad - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=66225938

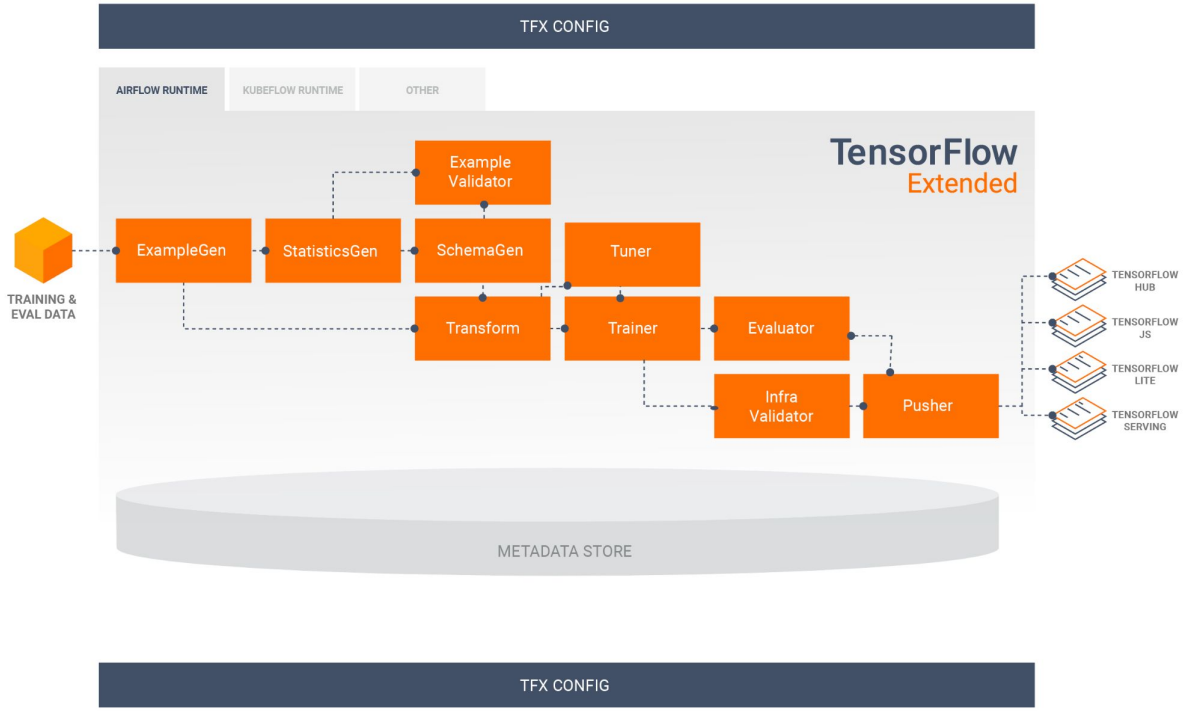


https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Czym jest *TensorFlow i TensorFlow Extended (TFX)*?

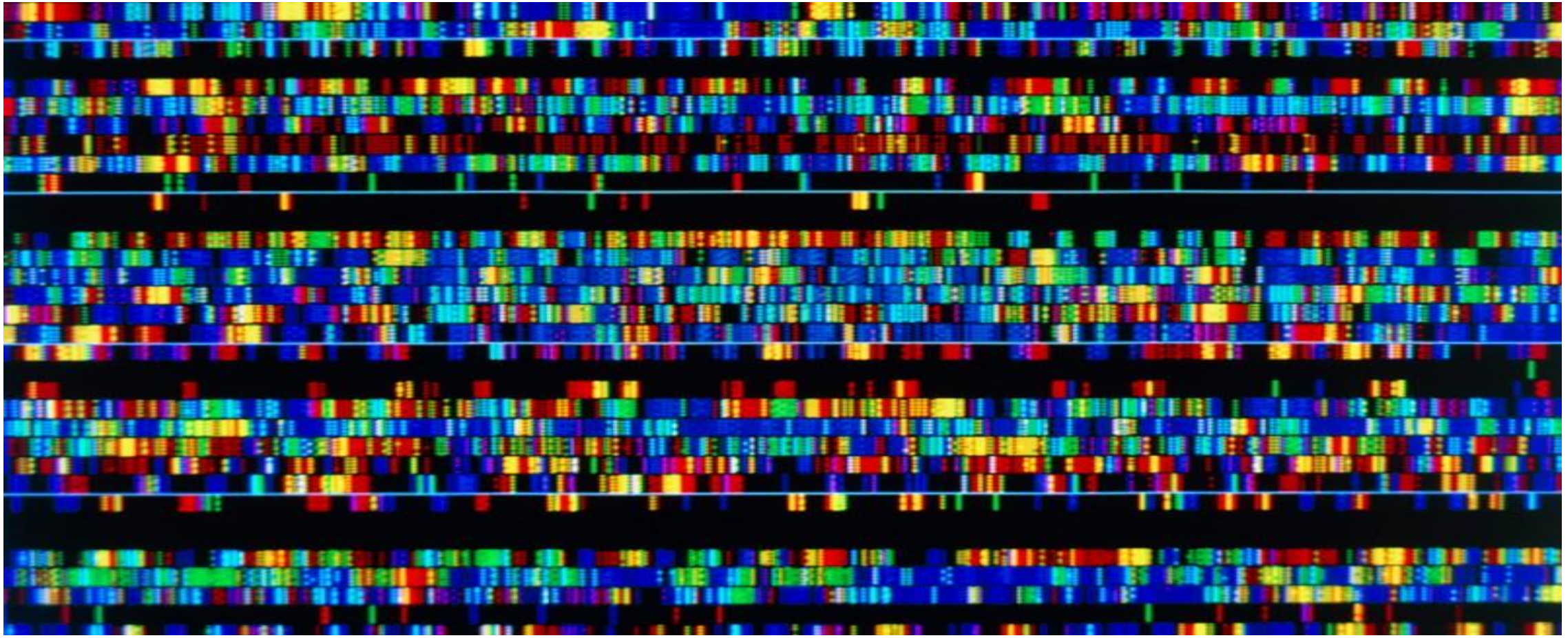
Jeśli jeszcze nie wiemy...

TensorFlow



Co robimy?

Sekwencje i rekomendacje



Jak dobre okazały się modele?

Szybki Benchmarking: wskaźniki Rec@5 i MRR@5 dla wybranej grupy produktowej

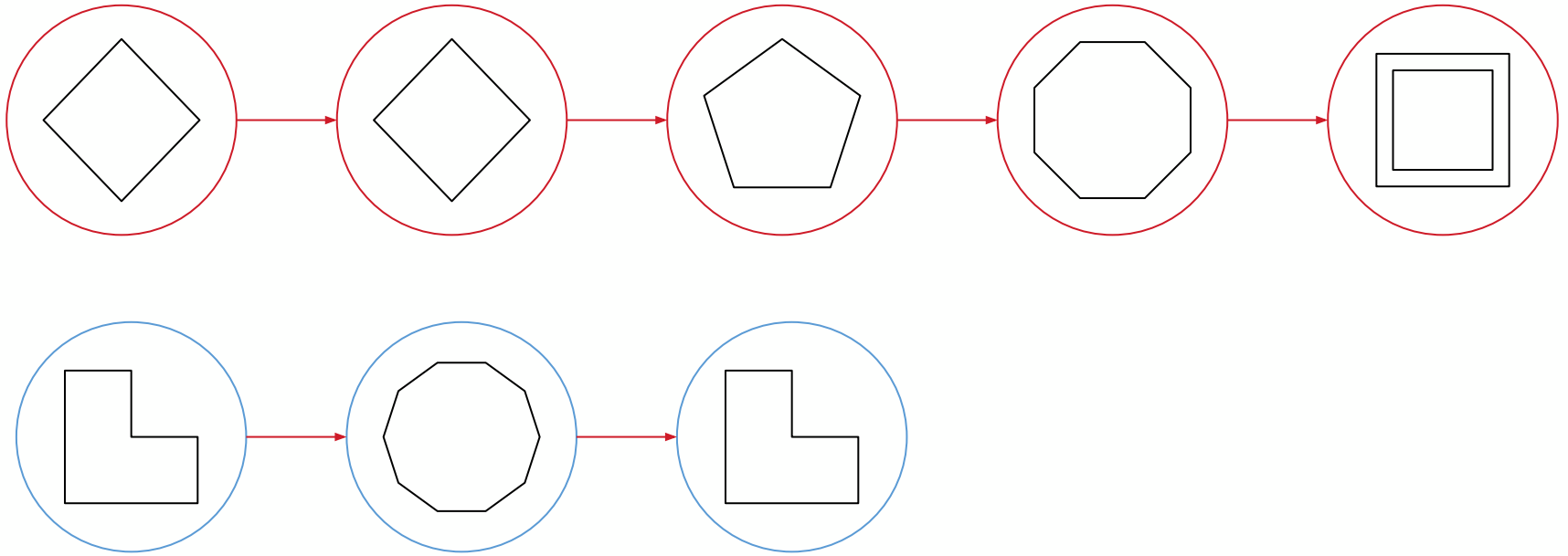
Metoda	Rec@5	MRR@5
VSKNN	0.398	0.508
Architektura Deep Learningowa #1	0.386	0.493
Architektura Deep Learningowa #2	0.371	0.488
Architektura Deep Learningowa #3	0.355	0.457
Architektura ML #1	0.341	0.508
Architektura ML #2	*0.165*	*0.210*

Rec@5: Recall at 5 -> Liczba poprawnie zidentyfikowanych produktów spośród 5 pierwszych rekomendacji; wyciągamy średnią dla wszystkich sesji.

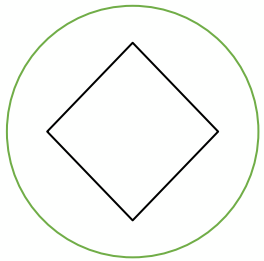
MRR@5: Mean Reciprocal Rank at 5 -> Odwrotność pozycji pierwszego istotnego produktu w sekwencji produktowej, wyciągamy średnią dla wszystkich sesji.j

k-NN bazujący na sesjach: czym jest sesja?

Dane wejściowe



?



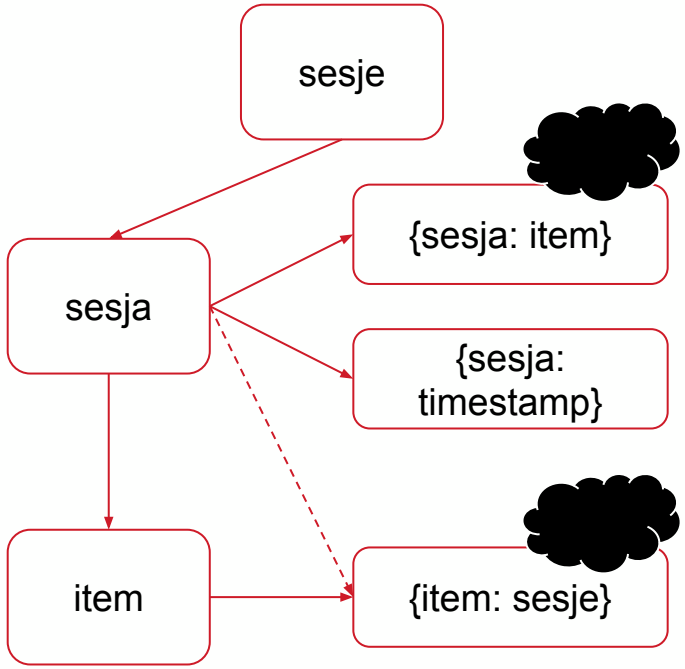
k-NN bazujący na sesjach: co zyskujemy?

1. **Prostota.**
2. **Solidność:** wyniki rekomendacji uzyskiwane przez algorytm są bardzo bliskie tuningowanym (mocno) sieciom neuronowym [1,2].
3. **Prędkość**: zmapowane sesje i produkty dają szybkie wyniki.
4. **Wzorzec do testów sieci neuronowych.**
5. **Ochrona przed nadmiernym dopasowaniem.**
6. **Wgląd w proces rekomendacji**.

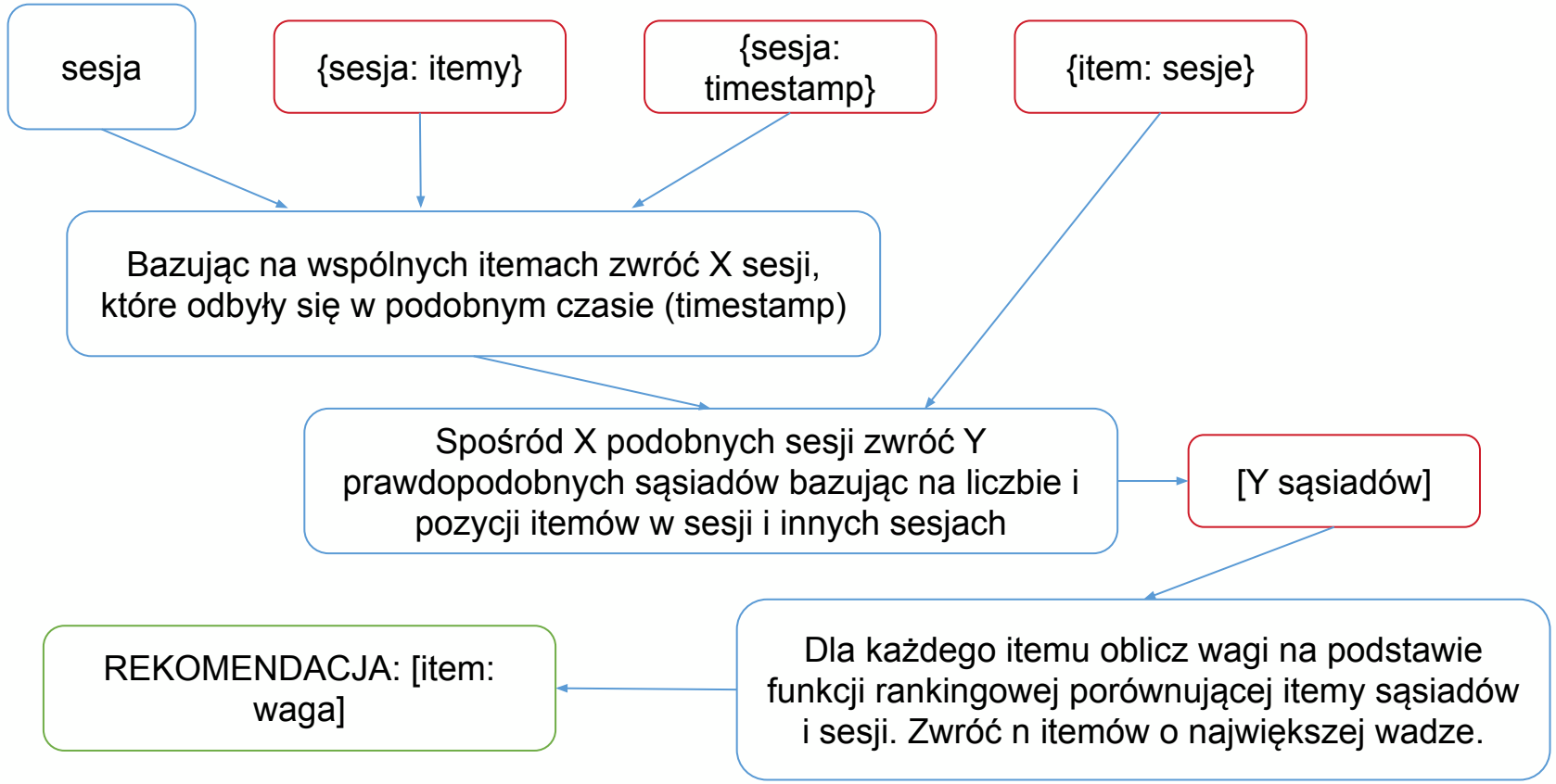
[1] Guo H., Tang R., Ye Y., Liu F., Zhang Y. (2019) A Novel KNN Approach for Session-Based Recommendation. In: Yang Q., Zhou ZH., Gong Z., Zhang ML., Huang SJ. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2019. Lecture Notes in Computer Science, vol 11440. Springer, Cham. https://doi.org/10.1007/978-3-030-16145-3_30
[2] Own research

k-NN bazujący na sesjach: jaka jest struktura algorytmu?

fit()



predict()



k-NN bazujący na sesjach: implementacja w Pythonie

Struktura Algorytmu: Słowniki, Zbiory i Listy

fit()

Tworzy mapy sesji i ich itemów (1);

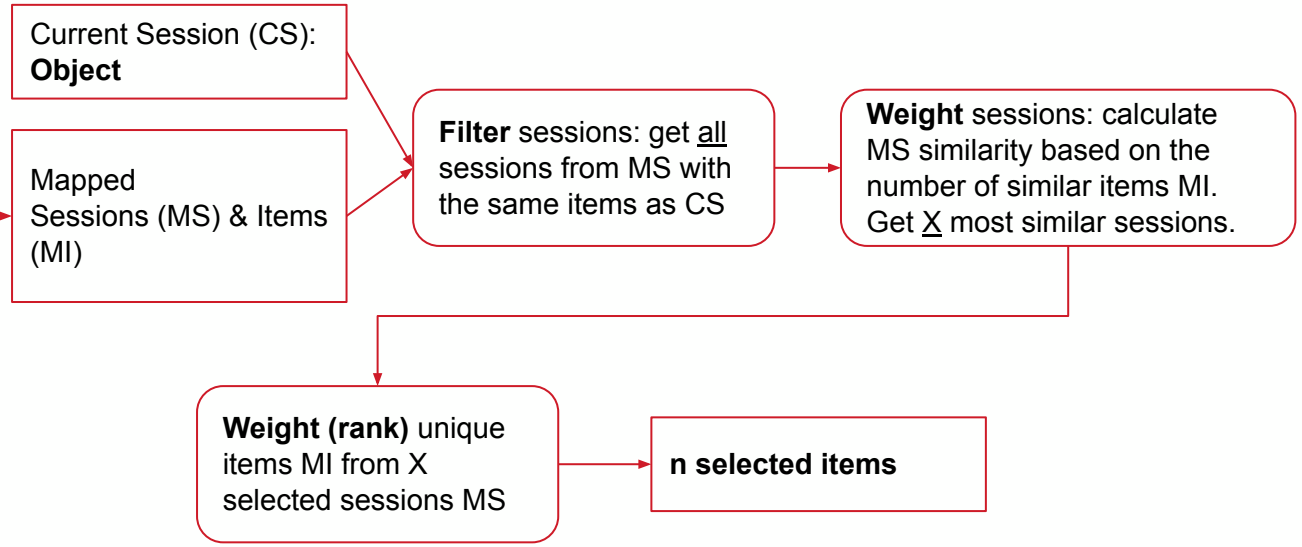
itemów i sesji, w których się one znalazły (2)

{session_id: { unique items }}

{unique item: { unique sessions with this item }}

predict()

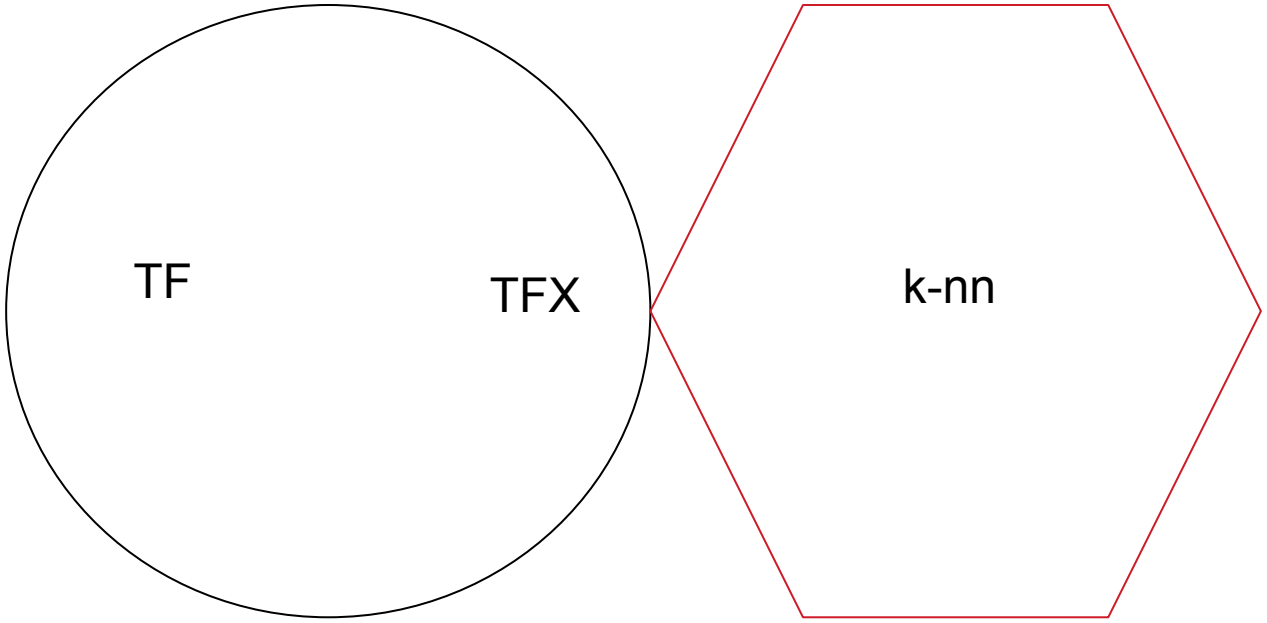
Oblicza dystans między produktami w danej sesji a **zapamiętanymi** sesjami i zwraca n rekomendacji



Dlaczego *ciśnienie* na TFX?

Implementacja w TensorFlow i TensorFlow Extended

1. **Uwspólnienie środowiska produkcyjnego**: podstawowy model oparty jest na architekturze sieci neuronowej i napisany jest w TensorFlow. Wszystkie modele referencyjne powinny być umieszczone w tym samym środowisku, by ograniczyć narzut pracy związany z przełączeniami między środowiskami.
2. **TFX == TF**: Modele napisane w TensorFlow działają poprawnie w produkcyjnym pipeline TFX, który jest rynkowym standardem.



Problem #1: kompatybilność TF i czystego Pythona

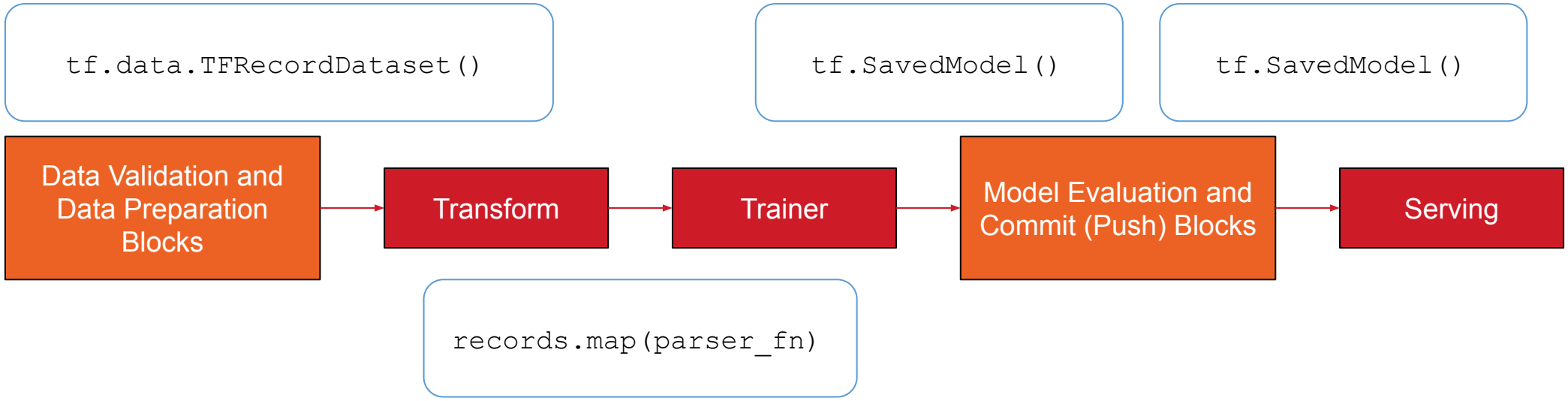
Co zgrzyta między Pythonem a TensorFlow? Z punktu widzenia implementacji algorytmu SKNN.

Python	TensorFlow
GIL (Global Interpreter Lock)	Graf (ang.: *graph*)
Podstawowe typy danych	Tensory (ang.: `Tensor`), Zmienne (ang.: `tf.Variable`)*
Słowniki	Brak odpowiednika w przełożeniu 1:1, trzeba operować na pozycjach w Tensorach
Elementy dynamiczne (np.: listy do których dodawane są elementy)	`tf.TensorArray(tf.string, size=0, dynamic_size=True)`
Elementy o zmiennych rozmiarach - np.: Słownik z listami o różnej długości	`RaggedTensor`
Dane wejściowe mogą być w różnych formatach**	Preferowany format `tf.Records`
(Domyślne) Parametry funkcji jako zmienne różnych typów	Parametry jako `tf.Constant`

*tf.Variable to Tensor ze zmiennymi elementami
**W granicach rozsądku, chodzi o wejście z płaskich plików albo baz danych

Problem #2: łączenie customowych modeli w pipeline TFX

Zmiana myślenia o modelu jak i o danych!



Implementacja w TFX: struktury danych

Implementacja customowego modelu w TensorFlow i TensorFlow Extended

session_id: {unique items within a session}

unique item: {unique sessions with this item}

`Tensor` Sesji

`RaggedTensor` Itemów w sesjach

`Tensor` Itemów

`RaggedTensor` Sesji z Itemami

Wszystkie stałe jako `tf.Constant`

Metadane modelu jak `tf.TensorSpec()`

Implementacja w TFX: Fit / Predict

fit()

Tensor ST with x encrypted sessions (as **string**)

Ragged Tensor RST with x rows with encrypted items (as **string**) : session ST[i] -> items RST[i]

Tensor IT with x encrypted items (as **string**)

Ragged Tensor RIT with x rows with encrypted sessions (as **string**) : item IT[j] -> sessions RIT[j]

Memorized Tensors

predict()

Trainer module with TFX specific **run_fn** function. To save trained model it is important to set **signature_dict**.

The general logic of prediction is the same as with the Python core structures. Differences are within an implementation in TensorFlow. It requires use of Tensors instead of simple containers.

`tf.SavedModel()`

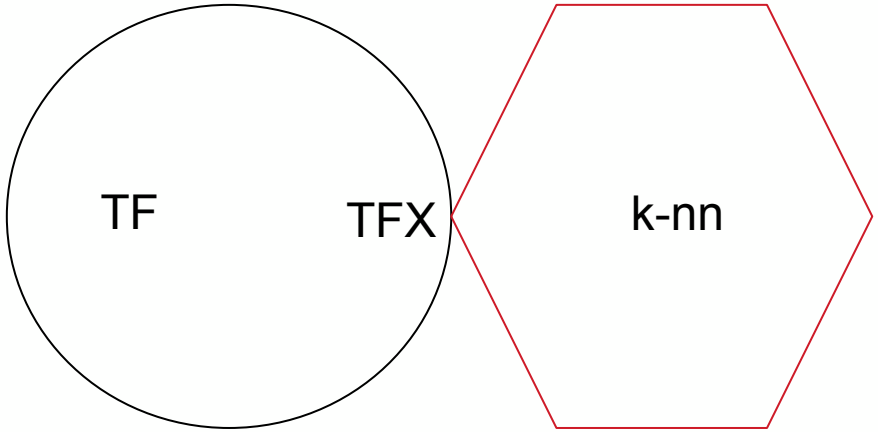
Implementacja w TFX: Przykład kodu

```
import tensorflow as tf
from tfx.components.trainer.executor import TrainerFnArgs
import tensorflow_transform as tft


# VSKNN Prediction model


class VSKNN(tf.Module):

    def __init__(self, session_items_rtensor,
                         session_items_sessions_tensor,
                         session_items_timestamps_tensor,
                         item_sessions_items_tensor,
                         item_sessions_rtensor,
                         sample_size=10):
        pass

    def rank_items_in_sessions(self, ranked_neighbors, ranks_of_neighbors, session, no_of_items):
        pass

    @tf.function(experimental_relax_shapes=True)
    def __call__(self, session, no_of_closest_items):
        session_neighbors, neighbors_ranks = self.nearest_neighbors(session)
        closest_items, items_ranks = self.rank_items_in_sessions(
            session_neighbors, neighbors_ranks, session, no_of_closest_items
        )
        return closest_items, items_ranks

    def nearest_neighbors(self, session):
        pass
```

```
def run_fn(fn_args: TrainerFnArgs):
    tf_transform_output = tft.TFTransformOutput(fn_args.transform_output)
    input_vsknn = InputVSKNN().fit(tf_transform_output, _BATCH_SIZE)
    vsknn_model = VSKNN(*input_vsknn, sample_size=_SAMPLE_SIZE)
    signature_dict = {
        "clickedItems": tf.TensorSpec(shape=[], dtype=tf.string, name="clickedItems")
    }
    model_path = fn_args.serving_model_dir
    tf.saved_model.save(
        vsknn_model,
        model_path,
        signatures=vsknn_model.__call__.get_concrete_function(
            signature_dict,
            tf.TensorSpec(shape=[], dtype=tf.string, name="PredictedItems"),
        ),
    )
```

Implementacja w TFX: co jest nie tak?

1. Zmienne rozmiary Tensorów.
2. Szeregi czasowe EWOLUUJĄ.
3. Zmiany w strukturze danych wejściowych.
4. Problemy z pamięcią przy wielkich zbiorach danych.
5. Złożoność kodu.



Q/A: Pytania i odpowiedzi

Github: **@SimonMolinsky**

Twitter: **@SimonMolinsky**

Linkedin: **@szymonmolinskipl**

Blog: **https://ml-gis-service.com**

Sales Intelligence: **https://salesintelligence.pl**