**Entity Matching**: Determine whether 2 or more records (represented by rows in a table or in different tables) are actually referring to the same real world entity. That problem might be significant in the task of data integration where properties might be consisted of large textual strings, which also provide a good opportunity for using NLP models to solve that.

**The goal of the paper**: Analyzing the components of pre-trained and fine-tuned BERT architectures applies to an EM task.

Past analysis of which knowledge is learned and applied by transformers-based architectures typically follow 2 main directions:

* Directly evaluate the contribution of specific architecture components (such as contextualized embedding or attention models)
* Examine the parameters of probing classifiers trained on top of the models

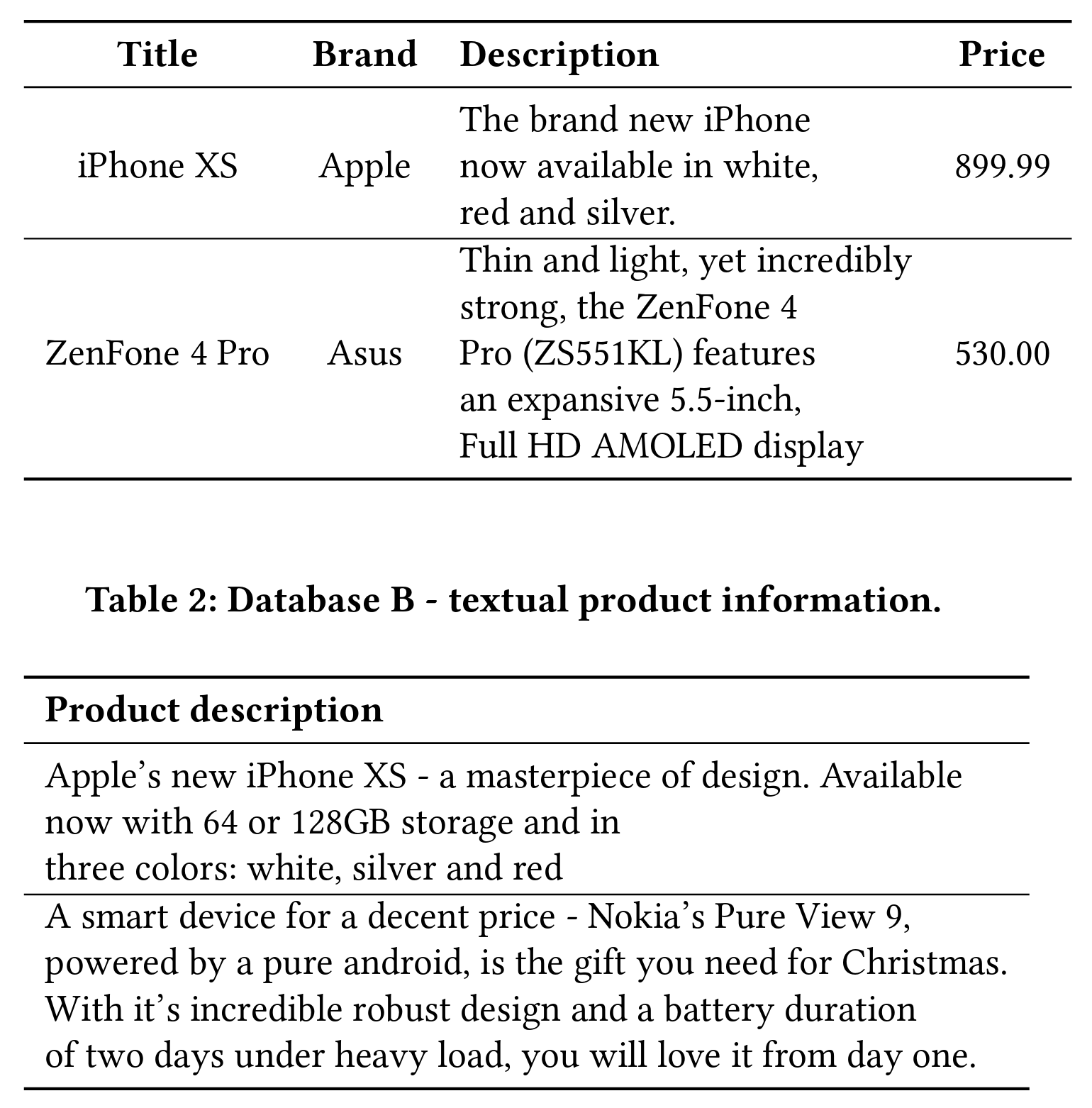
Inspired by these past analyses, the authors of the paper aim to inspect the ability of BERT-based approaches to preform EM, focused on the first direction.

Questions that provide guidance to the research:

* What is the impact of fine-tuning on the effectiveness of the EM task?
* What are the capabilities of BERT to detect & exploit the special structure of EM datasets?
* To what extent BERT-based EM models rely on the semantic similarity of pairs of tokens?

In the past, the task of entity matching relied on rule-based approaches and hand-crafted heuristics (for example the Jaro-Winkler distance, known to work well on person names), but now it is most commonly done using ML algorithms, both for structured and semi-structured databases.

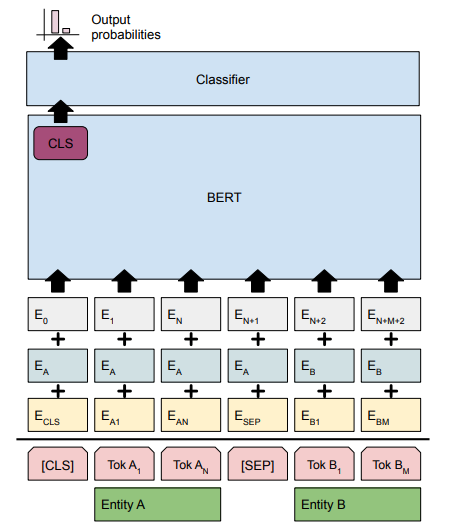
The problem today mainly occurs when the data consists of large textual instances, such as product descriptions, posts on Reddit, Quora, Stackoverflow and so on, and also when the data has structure, but attributes are dirty, e.g. the attribute "name" consists of a given name and a last name, while "given name" is empty. In those cases, traditional EM approaches provide only mediocre results or require large efforts in hand-crafted features. That is where recent advances in deep learning take the place for preforming the task.



**Cited Papers**

[4] - Entity Matching with Transformer Architectures - A Step Forward in Data Integration

This paper compares the performance of BERT, XLNet, DistilBERT and RoBERTa for the task of entity matching, both with and without fine tunning. It represents the approach of using the CLS token of BERT’s last layer in order to predict if it’s a non-match or a match



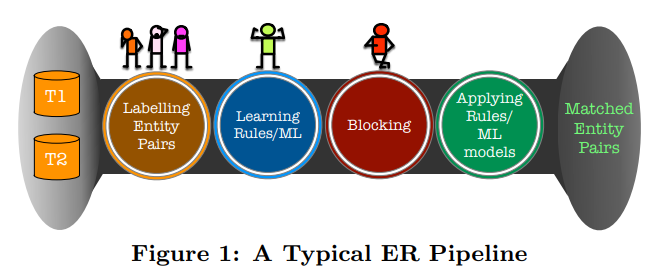
[7] - Deep Entity Matching with Pre-Trained Language Models

Here DITTO is presented, which is treated as the current state-of-the-art in out paper. It uses similar architecture presented in [4], but uses some optimizations like TF-IDF in order to eliminate noisy words, pre-processing step of blocking in order to prune pairs of entries that are unlikely matches to reduce the number of candidates of pairs to consider. It also uses domain knowledge to be injected into Ditto to emphasize what pieces of information are potentially important, and data-augmentations to make the model generalize and learn from harder samples.

[8] - Distributed Representations of Tuples for Entity Resolution

This article presents DeepER, which was a pioneer in using deep-learning (RNNs) in order to preform the task of EM / ER. Before using deep-learning, the task of ER was involving humans in every level:

1. Labeling entity pairs – human work needed to be done prior to training ML model or construction of rules according to these labels.
2. Learning rules or ML models using the labled data (unique for each dataset domain)
3. Blocking for reducing the number of comparisons
4. Applying the learned rules \ ML models



The idea suggested here is to use RNNs in order to get a better embeddings for a tuple (entity), taking into account the positional data of the tokens and the attributes (instead of averaging tokens representations (of word2Vec for example) in order to get attribute embedding, and then concat all the attributes embedding in order to get a tuple embedding).