University of Amsterdam
Applied Machine Learning Challenge

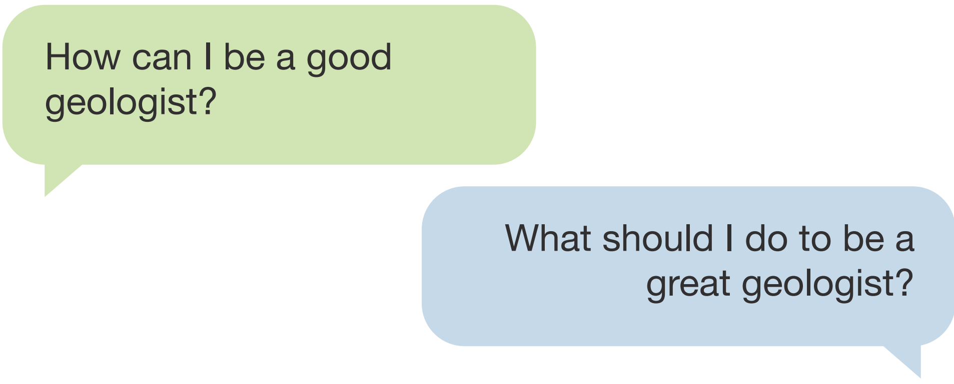
COME AGAIN?

Deduplicating Quora Question Pairs

Floris Hermsen & Jeroen Schmidt
December 2018

THE CHALLENGE

Quora is a community-based **question platform** where users can benefit from **knowledge sharing** by asking and answering each others' questions. Many questions are asked more than once, however, which poses an interesting **deduplication** challenge.



The **data set** for this challenge contains **323,164** annotated question pairs, of which **119,193** (36.9%) are marked as **duplicates**.

NETWORK ANALYSIS

Even though all question pairs are unique, many individual questions do appear **more than once** in both the training and testing data sets. We translated this **network** into a **graph structure** in which every node represents a question and every edge represents a question pair.

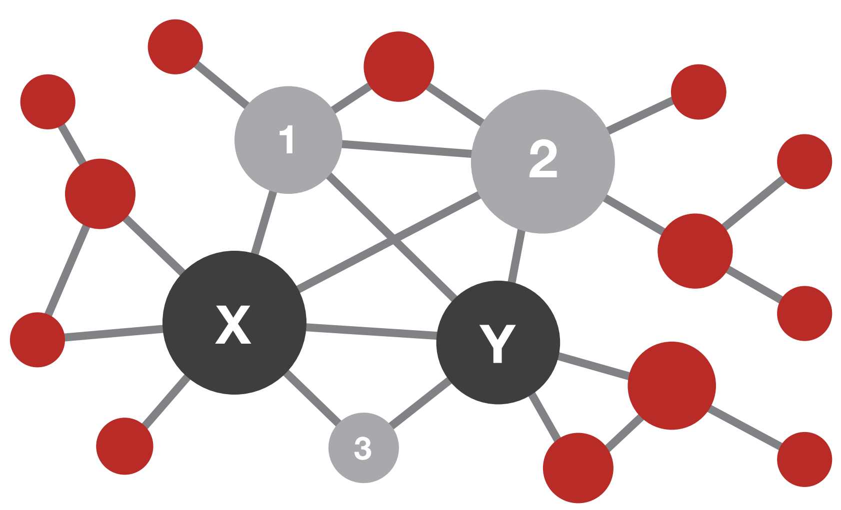


Figure 1. Nodes X and Y share 3 neighbours. This could indicate a higher duplicate probability.

Every question pair can now be assigned the following features: the **node degrees** of the constituent questions, their **intersection count** (shared neighbours) as well as mathematical **combinations** of these numbers.

LSTM ARCHITECTURE A

Our first **neural network** is trained on **word embedding** representations of the questions. It contains two **siamese LSTM** (long short-term memory) units, which receive the **unmerged** individual token embeddings, followed by different combination methods and multiple extra hidden layers. It was trained separately on both **word2vec** (Google News) as well as **GloVe** embeddings (Wikipedia).

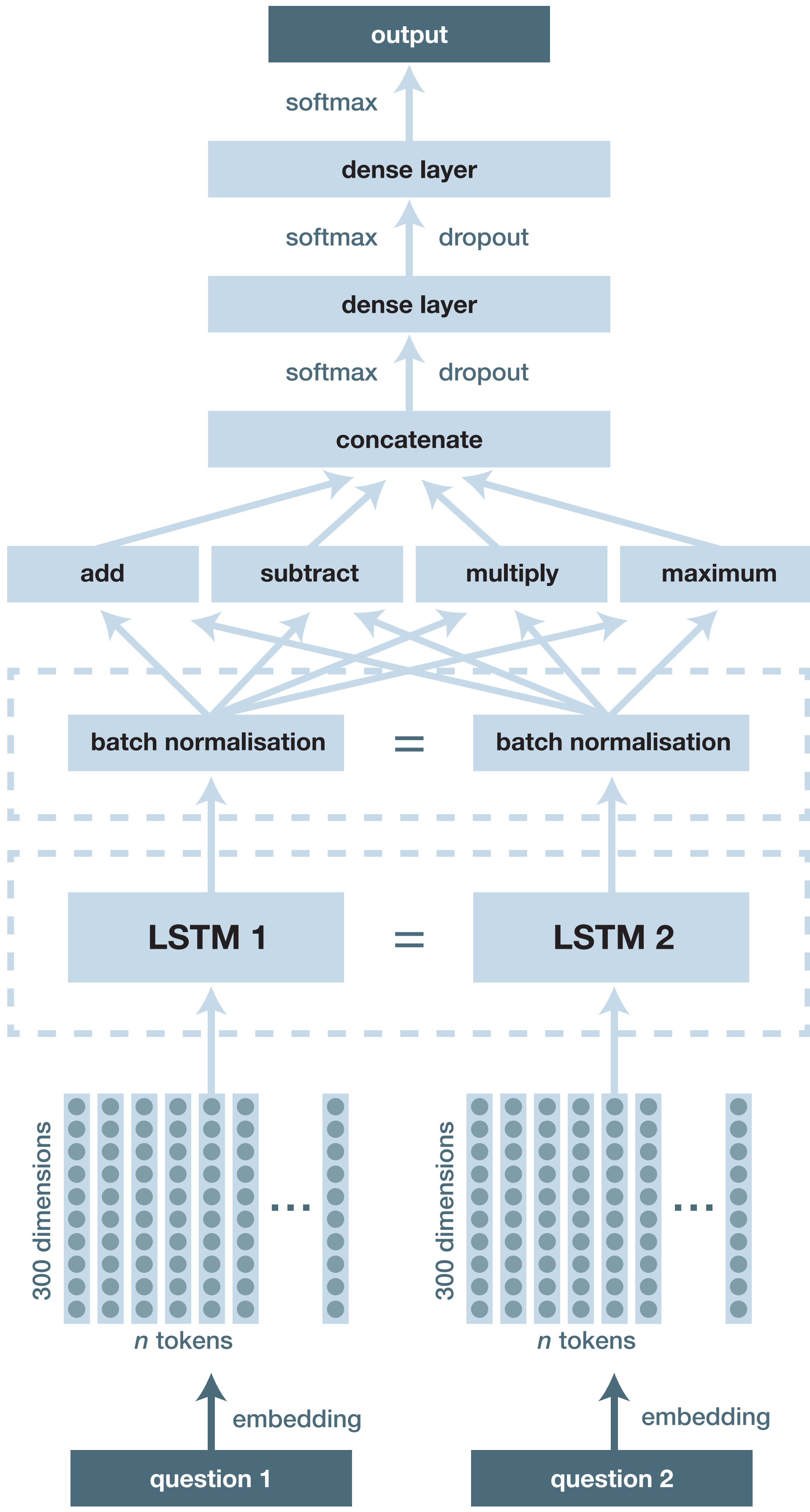


Figure 2. A schematic representation of network architecture A, containing a siamese LSTM unit, a set of combination methods and hidden layers. Dropout is employed to combat overfitting.

LSTM ARCHITECTURE B

Our second neural network is a variation on architecture A. In this case, the two outputs of the siamese LSTM units are **directly combined** into a single value using a **merge function**. The network was trained separately using two functions: the **dot product** and the **manhattan distance**, while each time using the **GloVe** embeddings.

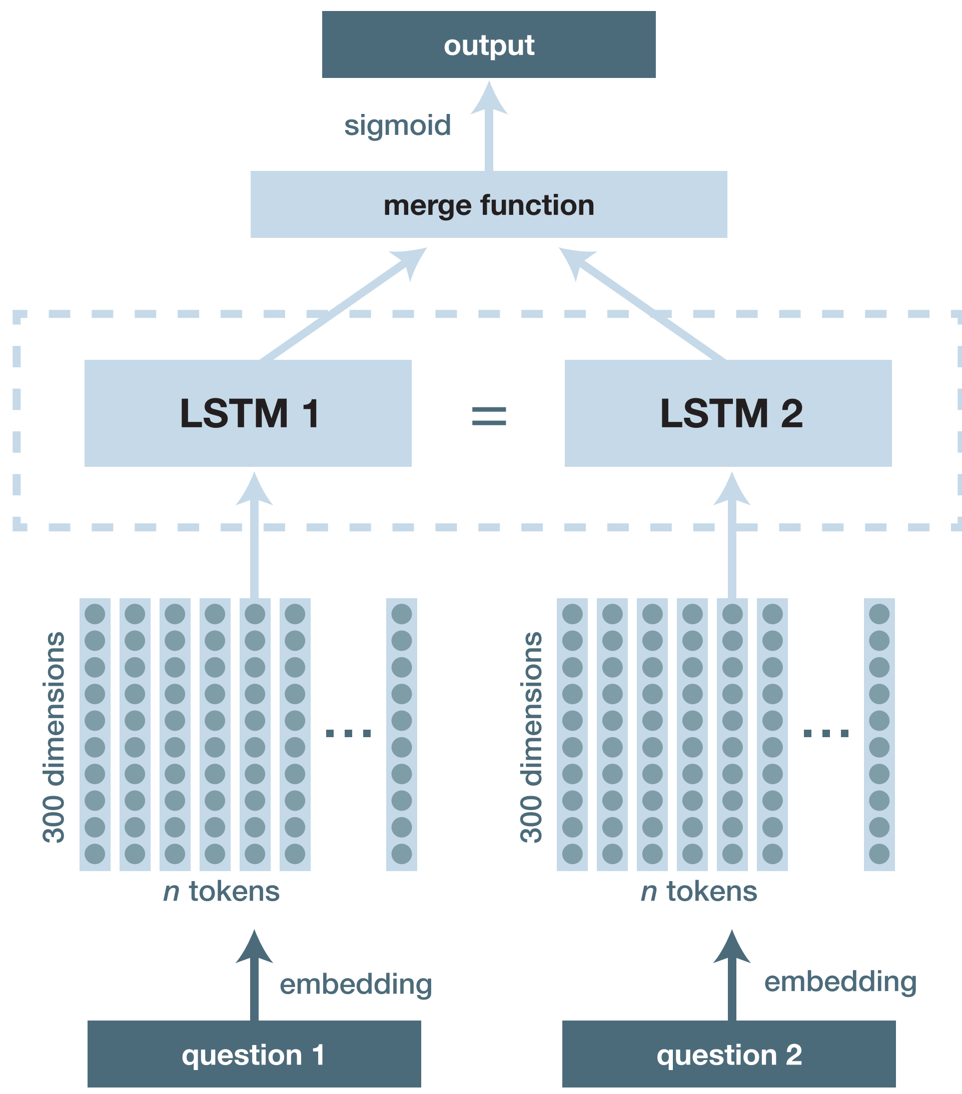


Figure 3. Network architecture B, containing a siamese LSTM unit, a merge function and a single sigmoid output node. There are no extra dense layers since the merging functions produce a single output value.

NLP FEATURES

In addition to the LSTM models and network analysis, we created **34 NLP features** for each question pair, including: **question length** comparisons, **token** comparisons (some **TFIDF**-weighted), **n-gram** comparisons, several averaged **embedding similarities** (some TFIDF-weighted), several **edit distances** and full token-by-token **embedding cross-correlation** scores. Also, a **named-entity-matching** score was computed (NER model from **spaCy**).

ENSEMBLE MODEL

Together with the generated graph and NLP features, the **output predictions** of LSTM architectures A & B (4 in total) serve as input for our ensemble model. Our final model was generated using **parallel GBDT** (gradient boosting decision trees), implemented using **XGBoost**.

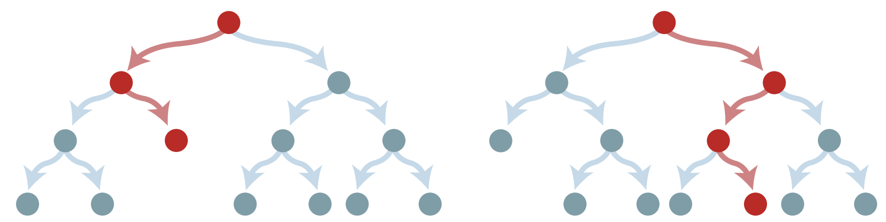
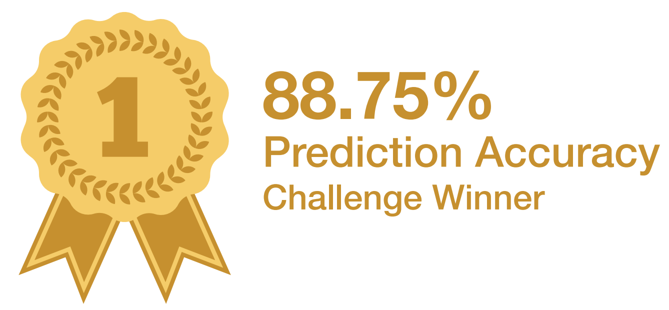


Figure 4. XGBoost generates multiple weighted decision trees based on the LSTM output predictions and the graph & NLP features, optimized based on the data set target varaiables and a loss function.

GBDT ensembles are very sensitive to **overfitting**. We therefore employed a **max depth** of **4** and **minimum child weight** of **10** based on cross-validation results.

RESULTS



Our **key finding** was to select sub-models not based on predictive performance but with a **validation loss comparable to the training loss**. This prevents the GDBT from **overfitting**.

The **TOP 5** most informative **NLP features** were: **BoW vectors** (TFIDF weights), **Levenshtein distance**, **embedding similarity** (TFIDF weights), **Jaro similarity** and the embedding **cross-correlations**.

Ensemble Model	**Accuracy**
Neural net (graph & NLP features)	86.89%
Neural net (graph & NLP features + LSTM output)	87.56%
XGBoost (graph & NLP features)	86.89%
XGBoost (graph & NLP features + LSTM output)	**88.75%**