SQuAD Project

Farinola Francesco

&

Vece Michele

Introduction

The aim of the project was to develop a Question Answering system to perform a Span Extraction task where given the context and the question, machine is required to extract a span of text from the corresponding context as the answer.

For the construction of our system, we were inspired by and used as baselines state-of-art works like DrQA and BiDAF.

The proposed architecture is a boundary model based on RNNs with a one-hop bidirectionctional attention mechanism using GloVe word embedding integrated with multiple granularity information.

Baseline (1) DrQA

DrQA system consists of two components:

1. Document Retriever module: for finding relevant articles
2. Document Reader: for extracting answers from a single document or a small collection of documents.

We focused on the Document Reader which exploits different input features for paragraph and question encoding like Exact Match, POS and NER tags, and term frequency.

Baseline(2) BiDAF

BiDAF model is a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity.

It includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation.

Data splitting

After checking correspondences between answers and spans, we decided to use as test set:

- rows containing multiple instances of the answer in the context, because the start index may refer to the wrong span;
- rows where the answer is not located at the suggested start index (if any).

Data are split in training and validation, according to the ratio 4/1. As suggested, the splitting is based on the title column, in order to maintain the questions/paragraphs regarding the same title in the same split.

Text cleaning

Contractions expansion: contractions are expanded (e.g., won't → will not)

Tokenization: the text is split in tokens. Spacy is used since it performs this operation in a more accurate way if compared to the split method of the class string;

Punctuation and symbols removal: non-alphabetic characters are removed.

Split of letters, digits and symbols: words containing numbers or symbols are split in more tokens (e.g., $5 → $ 5);

Spell correction: mispelled words are corrected by symspellpy.

Lemmatization: each word is lemmatized with the lemmatizer provided in nltk, in this way different inflected forms of the same word are reduced into the same root form;

Lowercase: uppercase characters are converted to lowercase in order to further normalize the text

Strip text: additional whitespaces are removed.

Embedding

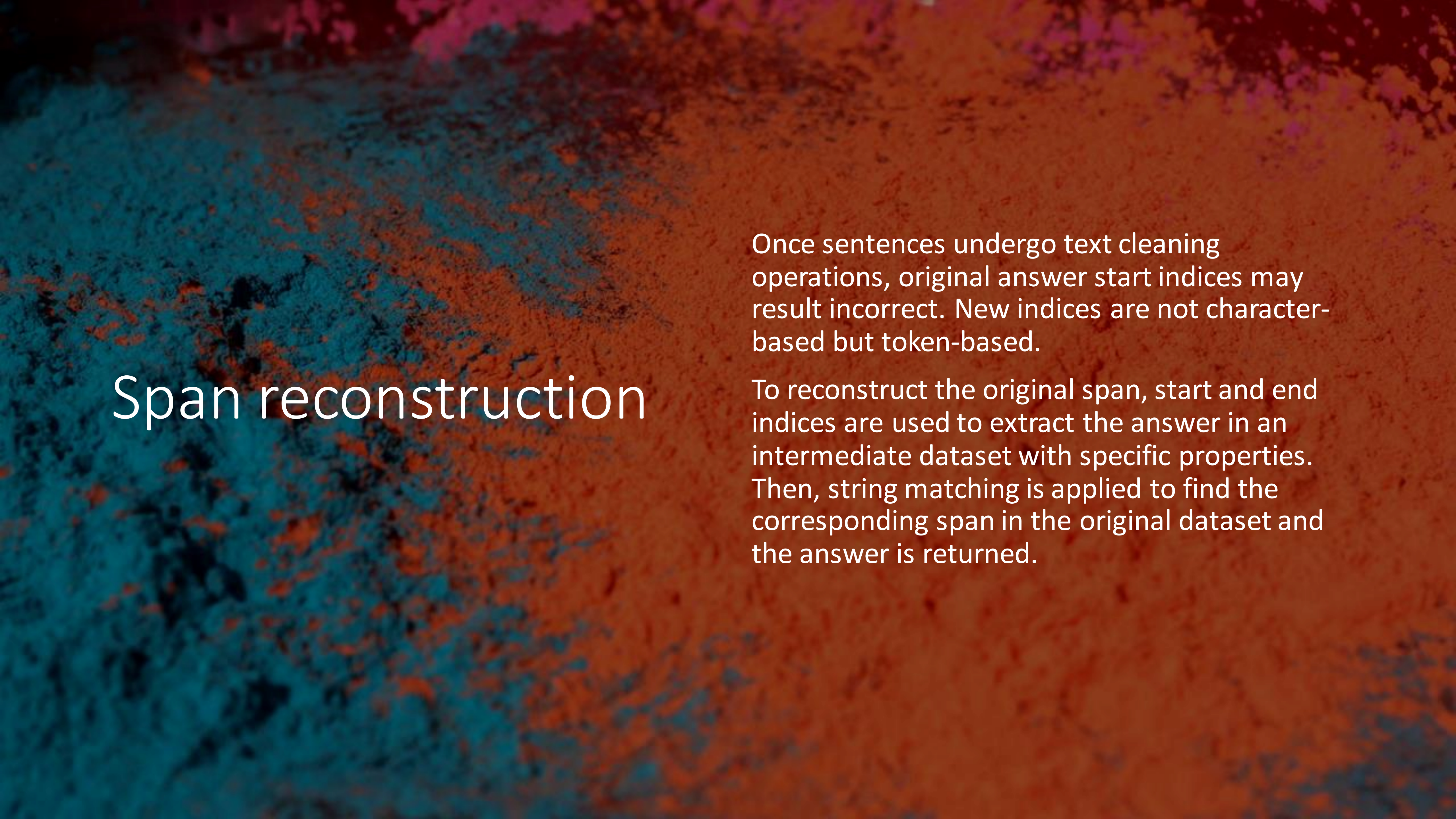
1. **Word embeddings:** GloVe word embeddings with dimension 100:
   - Terms in the GloVe vocabulary are assigned their GloVe vector representation
   - Out-of-vocabulary terms, are assigned the average of the vector representation of the nearby in-vocavulary terms. In case no neighbour is in the dictionary, a random vector is assigned.
2. **Character embeddings:** provided by chars2vec, with dimension 50.

   Represents each sequence of symbols of arbitrary length with a fixed length vector. Given two or more words, the higher the similarity in words spelling, the higher the similarity between the corresponding vectors.

Additional input features

1. **Term Frequency:** refers to the number of istances of the same word in the same sentence (context). It is achieved by using CountVectorizer from scikit-learn. The token pattern is "\S+" meaning everything but not the whitespace
2. **Exact match:** consist in finding which terms in the context match with at least one term in the question. We computed three types of exact match as binary features:
   - **Original match:** compares the term 'as they are' with basic text cleaning. Furthermore, we removed stopwords since they may cause noise and undermine the objective of exact match.
   - **Lowercase match:** as before, but also lowercase is applied
   - **Lemmatized match:** as before , but also lemmatization is applied
3. **POS tags:** obtained with spacy, which identifies 54 POS tags
4. **NER tags:** obtained with spacy, which identifies 20 NER categories. Also, a NONE tag indicating terms with no tag assigned.

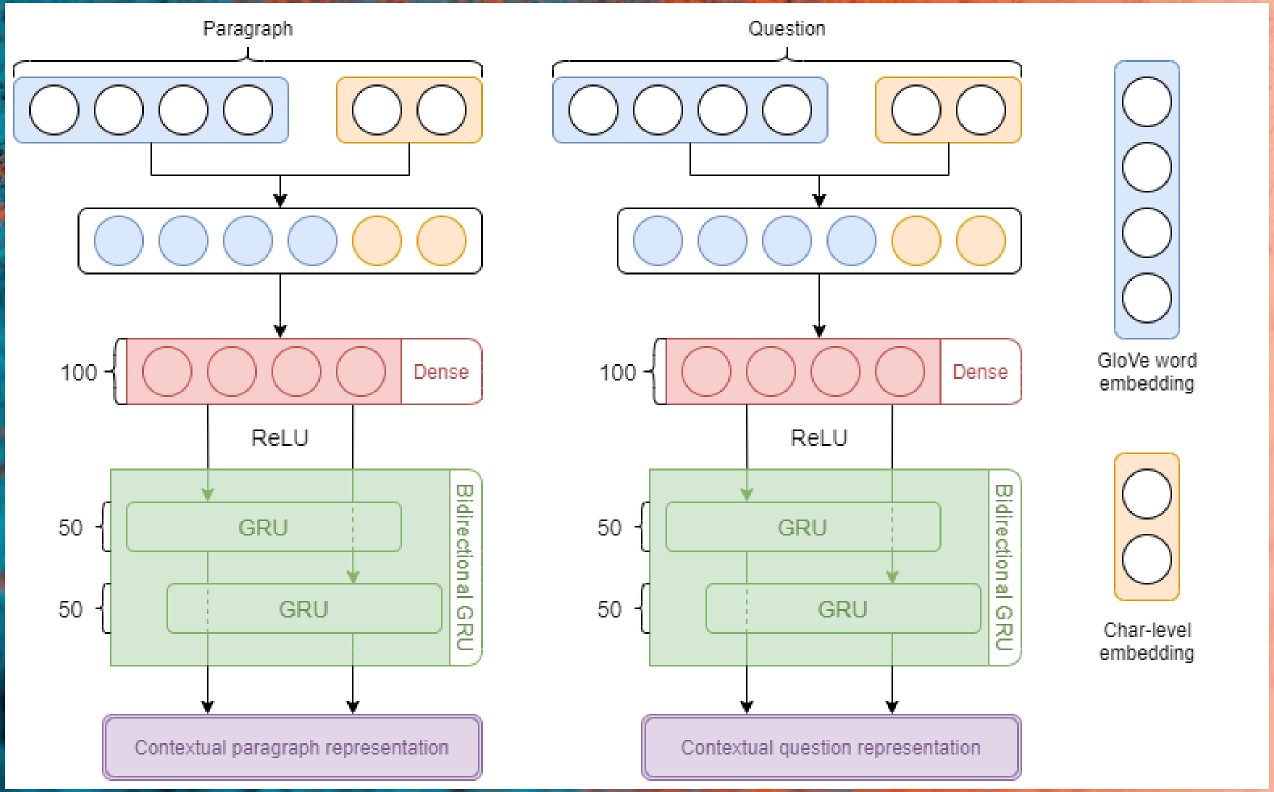
Both POS and NER tagging include a PAD token



Span reconstruction

Once sentences undergo text cleaning operations, original answer start indices may result incorrect. New indices are not character-based but token-based.

To reconstruct the original span, start and end indices are used to extract the answer in an intermediate dataset with specific properties. Then, string matching is applied to find the corresponding span in the original dataset and the answer is returned.



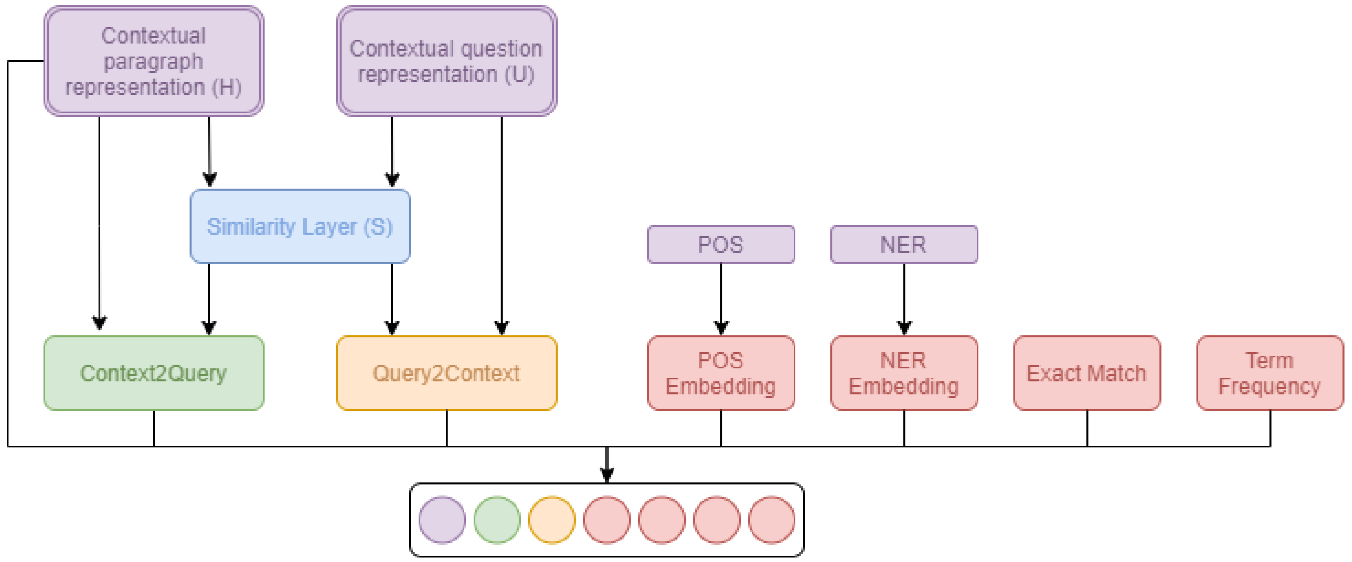
Model implementation

Contextual representations

- Context and question words are replaced by their GloVe word embedding and by their character embedding via two Embedding layers. These layers are then concatenated and passed to a Dense layer whose dimension is smaller than the sum of the two embedding dimensions.
- Outputs are then passed through a BiDirectional GRU to obtain contextual representations for contexts (H) and questions (U).

Model implementation

Attention and Merge



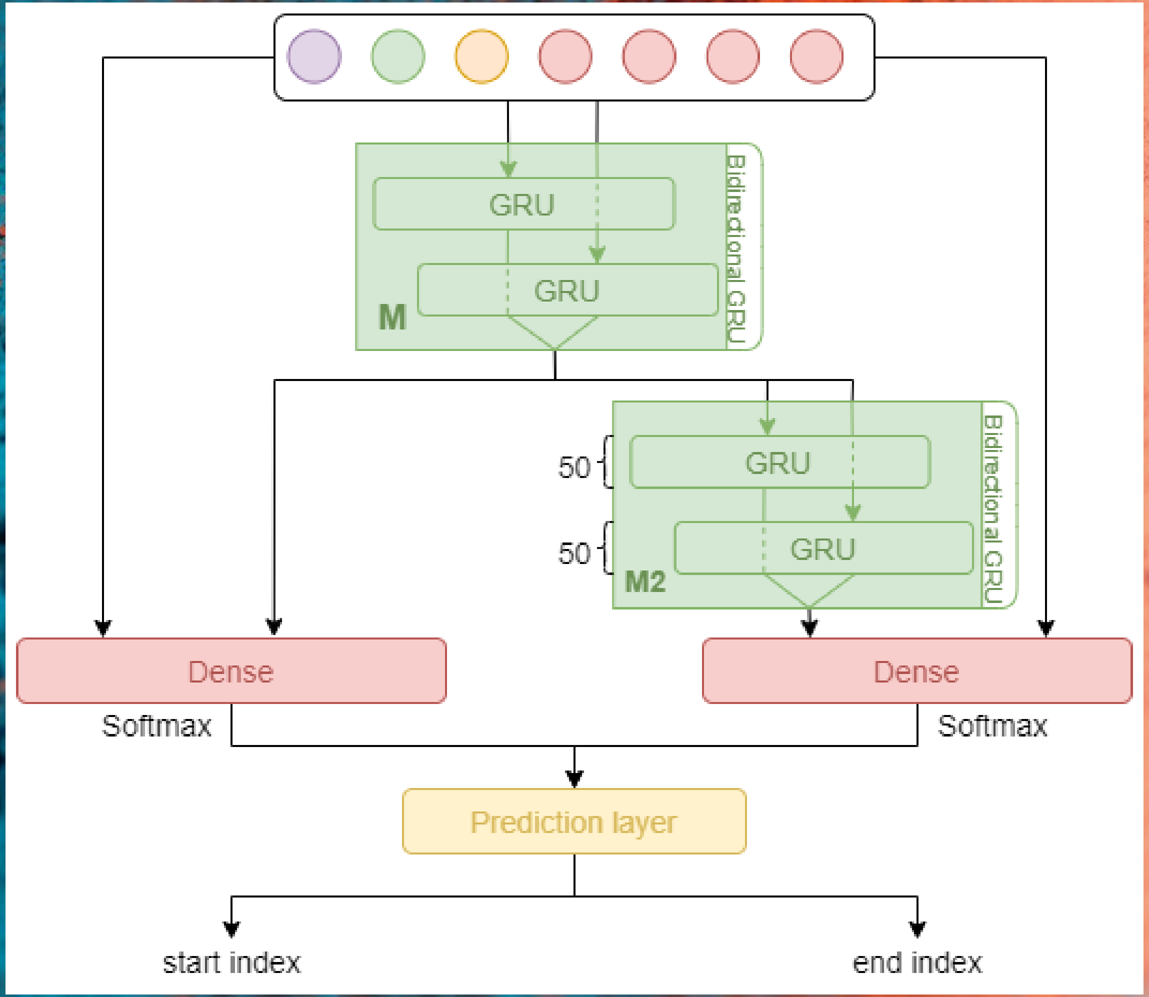
- Attention Layer:
  1. First, the similary matrix S is calculated. Each element si,j is obtained as: $s_{ij} = w^T[h; u; h \circ u]$
  2. Second, we compute the Context-to-Query attention that measures which query words are most relevant to each context, defined as:

     $$\tilde{U}_t = \sum a_{tj} U_j \quad \text{with} \quad a_t = \text{softmax}(S_t)$$
  3. Third, we computer Query-to-Context attention that refers to which context words have the closest similarity to one of the query word, defined as:

     $$\tilde{h} = \sum b_t H_j \quad \text{with} \quad b = \text{softmax}(\max_{col}(S))$$

- Merge Layer: we concatenate contextual embedding and attention vectors in the following way:
  - H, C2Q, H*C2Q, H*Q2C along with the additional input features

Model implementation
Modeling and Prediction layer

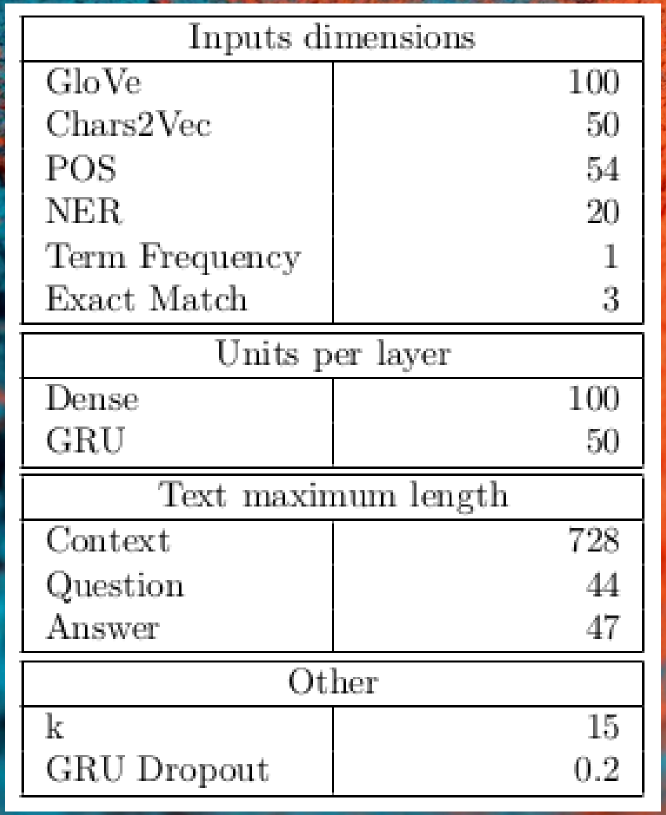


The output from the previous layer G is processed via GRUs (one for the start prediction, two for the end prediction), obtaining the modeling layers M and M2, respectively. They are concatenated to G, passed to Dense and TimeDistributed layers. After squeezing on the last axis, we get two tensors of dimension (context size), at which softmax is applied to return start and end probabilities for each context index.

The Prediction layer forces the following constraint inside the training of the model and not after, to compute the start and end indices pf the answer:

$$i_{start} \leq i_{end} \leq i_{start} + k$$

Final configuration

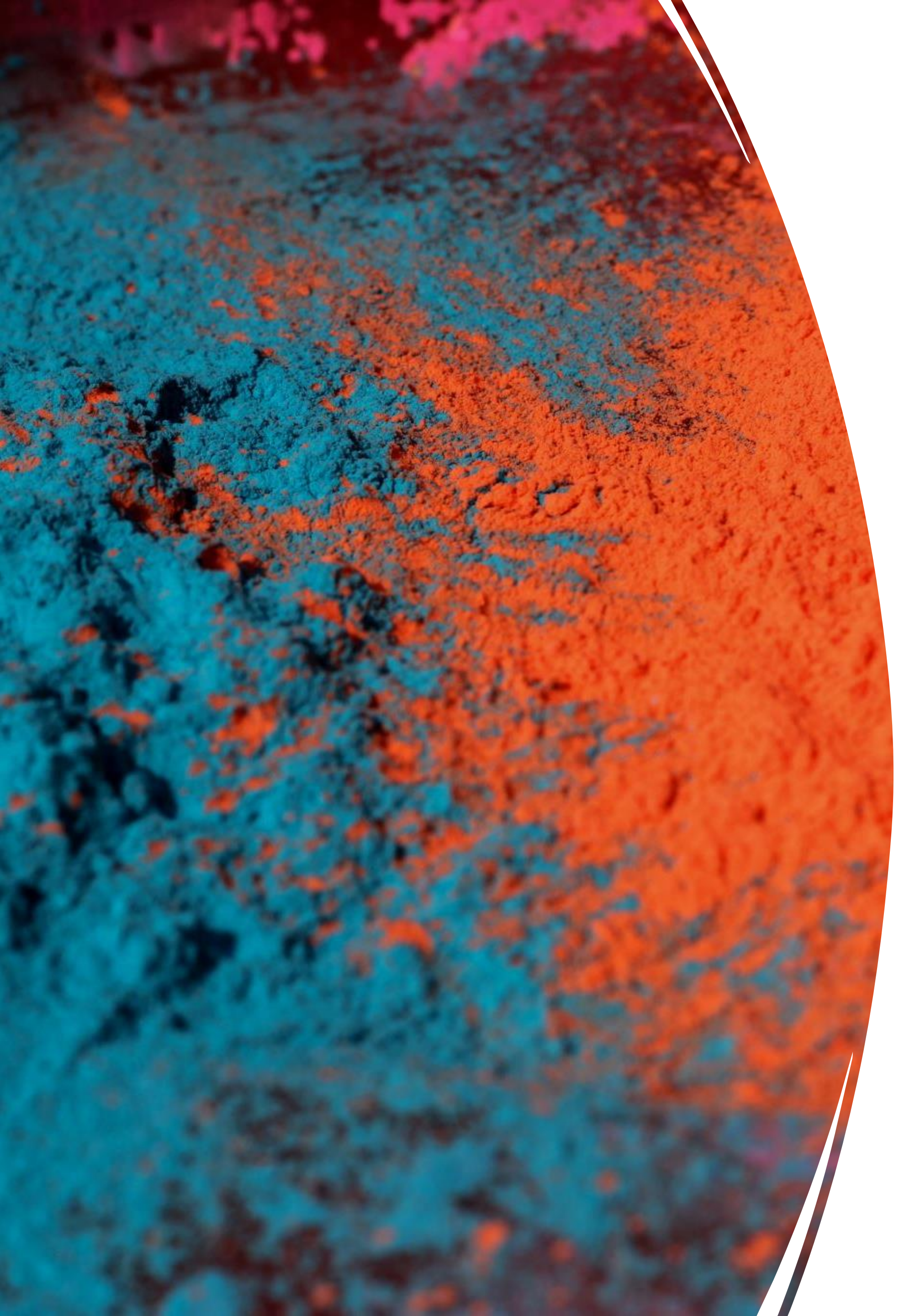


Inputs dimensions	
GloVe	100
Chars2Vec	50
POS	54
NER	20
Term Frequency	1
Exact Match	3
Units per layer	
Dense	100
GRU	50
Text maximum length	
Context	728
Question	44
Answer	47
Other	
k	15
GRU Dropout	0.2

The system was trained with Nadam optimizer, batch size = 16 and early stopping with patience = 5 on validation loss. The training stopped after 12 epochs.

Other configurations were tried with greater word and character embedding size, more dense and GRU units, and higher dropout rate.

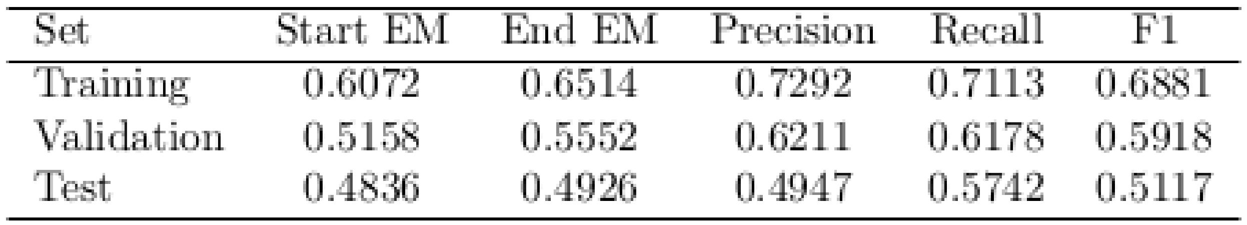
Also, the size of k was increased.



Results

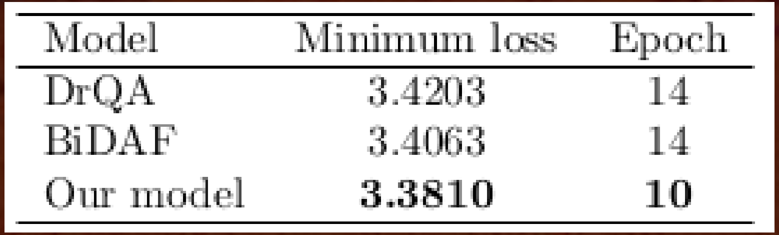
Running the evaluate.py script on the whole dataset, the model we developed reached an Exact Match of 52.27 and and F1 score of 67.44

Instead, the following table resumes metrics written by us, separately computed on training, validation and test set:

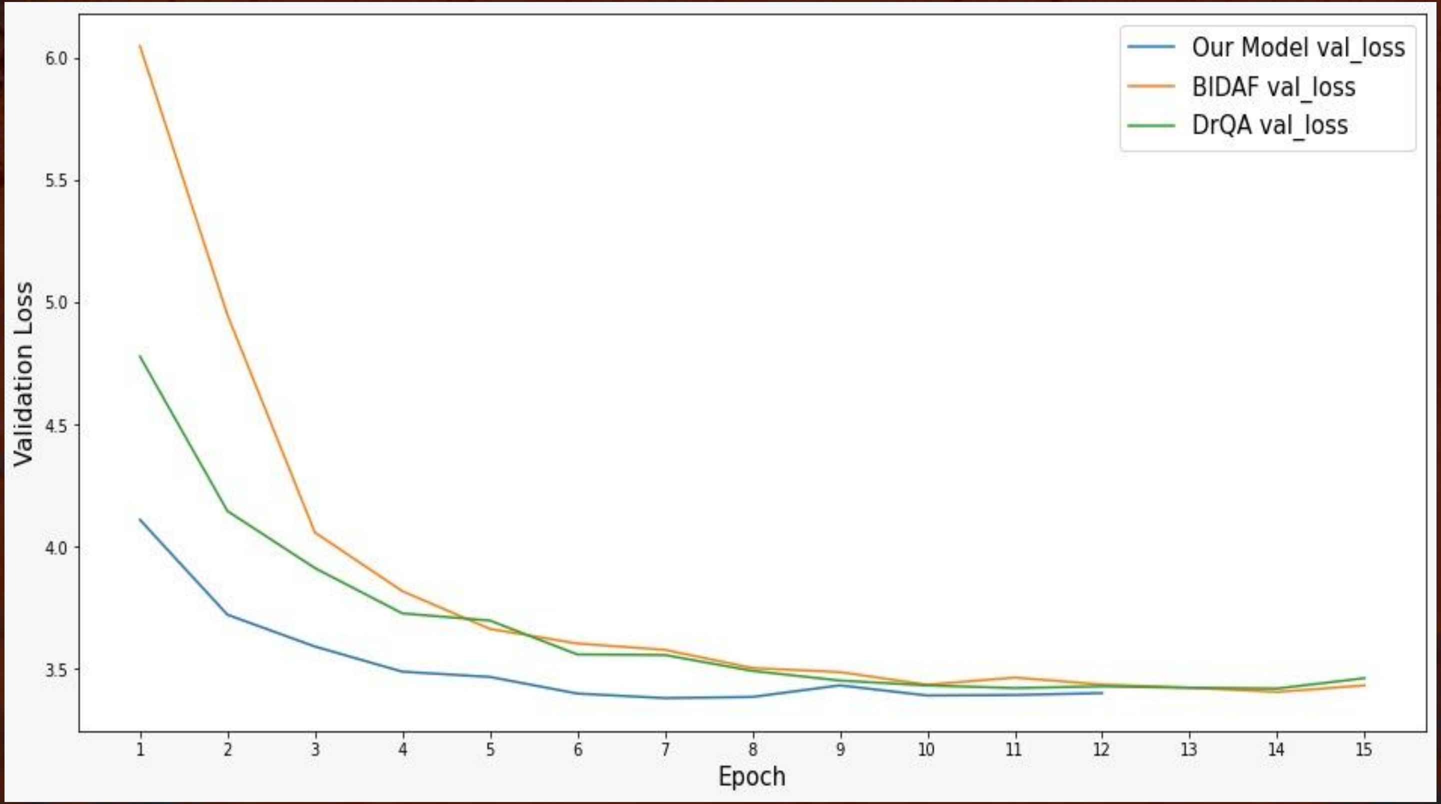


Set	Start EM	End EM	Precision	Recall	F1
Training	0.6072	0.6514	0.7292	0.7113	0.6881
Validation	0.5158	0.5552	0.6211	0.6178	0.5918
Test	0.4836	0.4926	0.4947	0.5742	0.5117

Comparison with baseline models



Model	Minimum loss	Epoch
DrQA	3.4203	14
BiDAF	3.4063	14
Our model	**3.3810**	**10**



Analysis of errors and considerations (1)

Comparing the predictions on the validation set to the expected answers, it is possible to notice that 41.20% are correct while we need to make some considerations on the remaining 58.80% (10007 wrong predictions).

Among the wrong predictions, we can identify some answers in which the model predicts correctly the amount or the date but misspells the answer (3.73% of the total predictions)

i.e.: How many copies did the album sell?

Predicted: 310,000	True: 310,000 copies

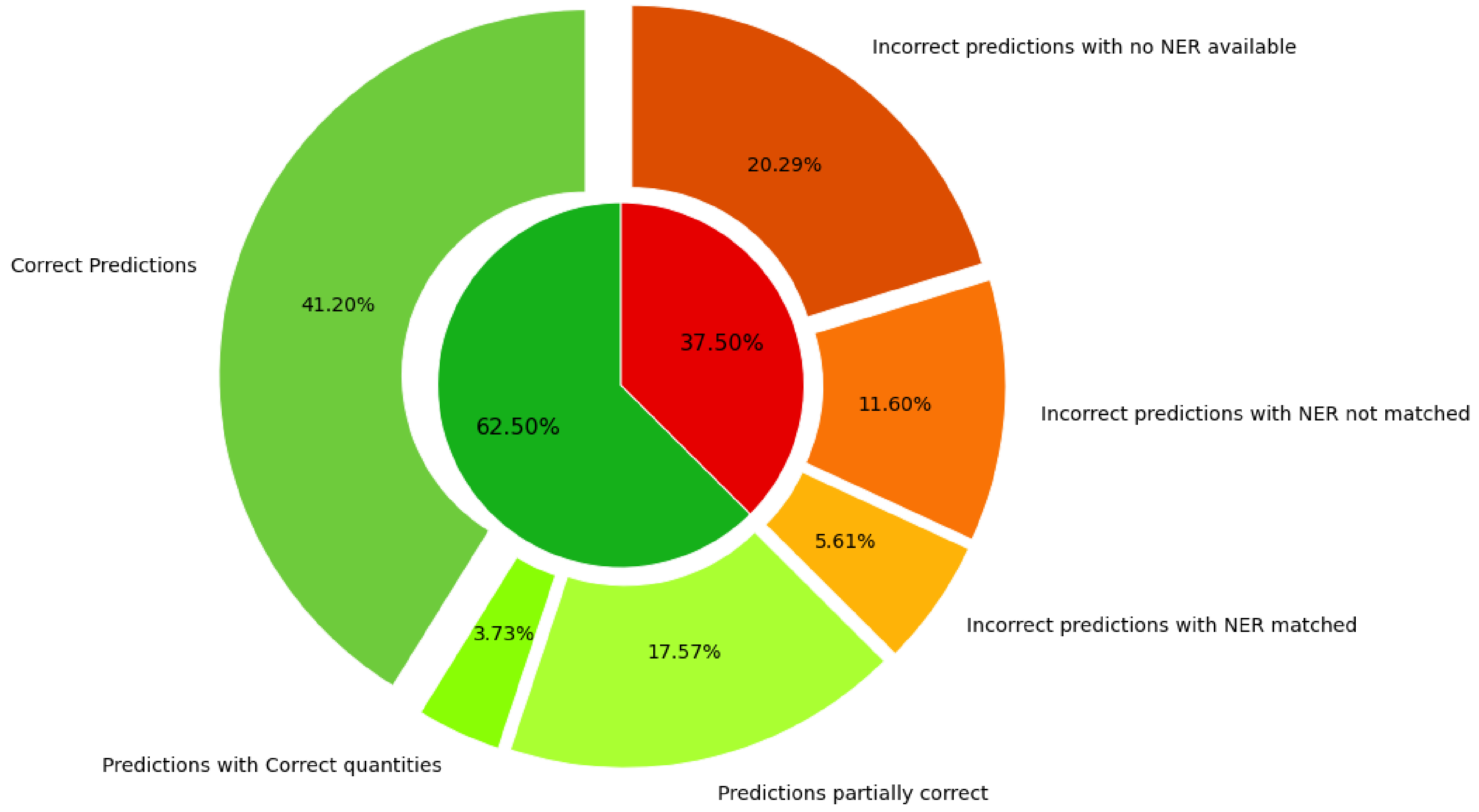
I.e.: When did she appear on the cover of GQ? (Beyoncé)

Predicted: January 2013	True: 2013

Furthermore, 17.57% of the predictions contain at least two words that are also contained inside the true answer (partial string matching).

Analysis of errors and considerations (2)

- If we compute NER for the predicted answers and the true ones, we notice that among all the incorrect answers, only 2929 provide a Named Entity category and in 950 of those, the category is predicted correctly.
- So, even if the prediction is incorrect, the system is often able to understand the kind of answer required.



References

- [1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. arXiv:1704.00051v2, 2017.
- [2] Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. Neural Machine Reading Comprehension: Methods and Trends. Applied Sciences, 9(18):3698, 2019.
- [3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bi-Directional Attention Flow for Machine Comprehension. arXiv:1611.01603v6, 2018.