Master in Artificial Intelligence
Human Language Engineering

Classification of Biomedical Questions from the BioASQ Challenge - Task 6b

Practical Work Report

December 29, 2019

Albert Espín
Lavanya Mandadapu

Contents

1 Introduction

According to its original formulation, the BioASQ Challenge - Task 6b [1] is a Question-Answering competition in the biomedical domain. The organizers provided a set of questions, divided in two sets, training and test, with the latter being labeled with a specific question type (or class), as well as an answer. The competitors were asked to build a Natural Language model to produce answers for the questions, in different batches, with increasingly more complex requirements, ranging from very short answers (e.g. as a response to a factoid question) to well-structured human-resembling answers that, ideally, should pass the Turing Test [2].

In the context of the Human Language Engineering course, a practical work is carried out, comprising the development of tasks that use the textual data from the BioASQ challenge, but are focused on the classification of biomedical questions, instead of the generation of answers to those questions. Namely, the work is divided in two tasks:

- **Task 1**: Development of a classifier of biomedical questions from the BioASQ Challenge - Task 6b, into 4 question types:
  - Factoid, i.e. a question that asks for a short, direct fact answer (e.g. when did some event take place).
  - List, i.e. a question that requests the enumeration of certain entities.
  - Summary, i.e. a question to obtain an explanation about the asked topic or matter.
  - Yes/No, i.e. a binary question with only two answer options, either to confirm (or validate) or deny (or refute) the asked predicate.

  The main goal of this part is the maximization of the classifier's accuracy, that should be as high as possible for all classes. Particularly, different metrics will be used to assess the quality of the results, presented in Chapter 4.

- **Task 2**: Co-training with an extensive broad-domain data set of 404,302 Quora question pairs to improve the classification accuracy, since the data of Task 1 is comprised only of a small number of questions (2252). Nevertheless, the Quora questions need to be filtered and classified first, before being usable in the model.

  In this task, the ultimate goal remains the same of Task 1, to maximize the accuracy of classifying the original questions; the difference is that the additionally described data is used to strengthen the model training process.

A secondary goal present in both tasks is to study and compare how different models can be applied for a same task, analyzing their strengths and weaknesses.

Before building any model, a data analysis is performed, as explained in Chapter 2 of this document, to examine the linguistic properties of the BioASQ questions data set, in the attempt of inferring which linguistic and Machine Learning techniques would potentially be the most effective to tackle the classification tasks. The linguistic, algorithmic and Machine Learning-driven approaches used to solve each challenge are expounded in

Chapter 3, while the comparative results of analyzing the different models is examined in Chapter 4. As a last step, Chapter 5 outlines the conclusions of this work and proposes directions for hypothetical future work on the subject of classifying biomedical questions.

2 Linguistic Data Analysis

When analyzing the data of a classification problem of supervised learning, one of the first questions that arise is whether the problem is balanced or unbalanced, i.e. whether all the expected classes have a similar number of instances or there are noteworthy irregularities in the class distribution. If the problem was unbalanced, it would be appropriate to study techniques to overcome such scenario, since the loss functions of Machine Learning algorithms can easily be prone to ignore or give less importance to making errors in classes with a minority representation. Possible solutions for that scenario would include weighting the penalization of each class in an inversely proportional way to their number of instances to compensate the under-representation of minority classes, or to perform data augmentation, by synthetically creating new instances for the classes with less examples.

Figure 1 illustrates the class distribution of the questions of the BioASQ Challenge that need to be classified in the tasks. It can be observed that the number of classes for all classes are similar, even if not perfectly balanced: the most numerous question type is Factoid, with 619 instances, almost the same as Yes/No, that has 616, whilst Summary has 531 and List has 485. Since we are not facing an unbalanced distribution, it is not expected that the measures proposed in the previous paragraph will be needed (even though the data will be augmented with Quora questions in Task 2, but because the overall number of questions is small, not due to unbalancing issues). Nevertheless, it will be interesting to analyze whether the accuracy of models for the class with less instances (List) is clearly lower than that of the most extensively represented class (Factoid), even if our initial hypothesis is that significant differences should not appear.

When building a Natural Language Understanding model, it is crucial to determine the complexity of the text that will be processed by the model. Textual complexity can be measured in multiple levels, from lexical (at word level) to pragmatic (with global contextual meaning of the text), including in the middle the syntactic and semantic levels of words, as well as locally adjacent groups of words. A first step to study the complexity of texts is to measure the number of tokens (individual words or independent linguistic characters, such as question marks) contained in every word. To perform this task, the NLTK [3] word tokenizer was used, due to its free availability and reputation of being considerably accurate. Figure 2 presents the distribution of token number among the BioASQ questions. It is easy to see that the distribution resembles to Gaussian distribution. In this case, one can see that all texts have at least 3 tokens and no more than 33, but the vast majority of the texts, 1881 of the 2252 (83.53%) have between 6 and 14 tokens; if we want to expand the range to cover a higher percentage of the data, we can see that 2190 questions (97.25%) have between 4 and 18 tokens. In all these cases, the questions are quite short, although not extremely short on average.

Therefore, the complexity of the texts is expected to be quite simple (but not trivial), since with such moderately small number of tokens it is not possible to have many sentences (most questions are expected to have a single sentence, instead), with complex relations between them, including phenomena such as coreferences that would need to be resolved at pragmatic level to gain a full understanding of the question's meaning.

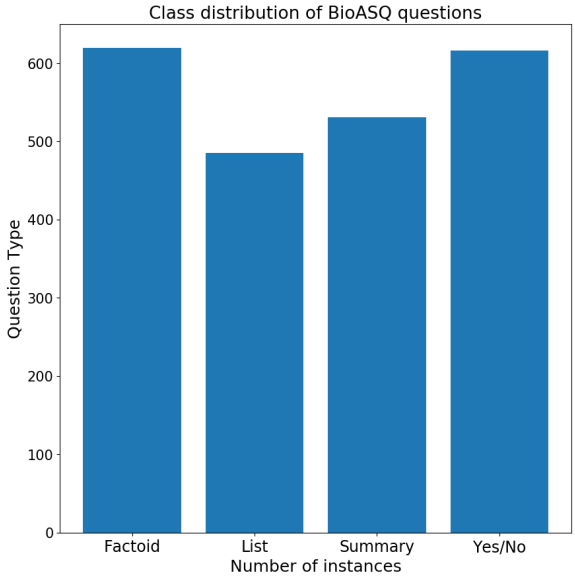

Figure 1: Question type distribution of the questions of the BioASQ Challenge.

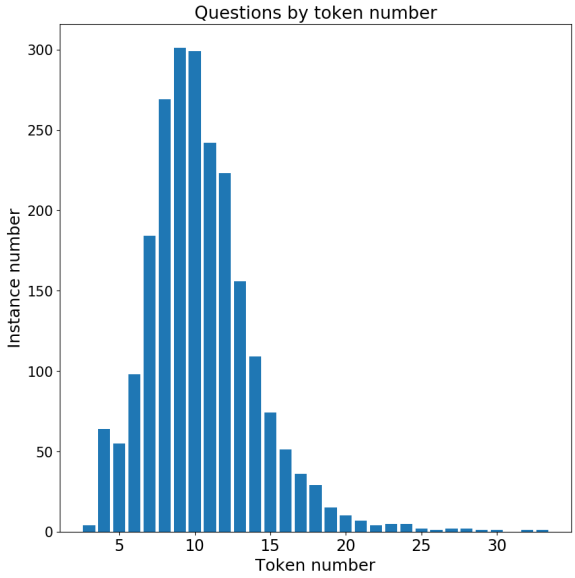

Figure 2: Distribution of the number of tokens among the BioASQ questions.

To confirm the hypothesis that most questions have a single sentence, NLTK's sentence tokenizer, used at an intermediate step of word tokenization (sentence splitting), is applied. Validating our thoughts, it is found that 2238 questions have a single sentence, 12 have 2 sentences and 1 question has 3 sentences. These results are shown in Figure 3, but it should be noted that logarithmic scale was used, since the differences in magnitude were so high that otherwise it was very difficult to see the questions with 2 and 3 sentences. Since the texts in this problem mostly consist of a single moderately short sentence (not long on average), it seems clear that resolving problems that occur in multi-sentence problems, such as coreference resolution, would not give a significant advantage in classification terms. Instead, lexical and semantic information is bound to be useful, since interpreting the meaning of words in the sentence context can convey to understand the question type; as well as syntactic information, since the structure and phrase hierarchy of questions might be indicative of their type.

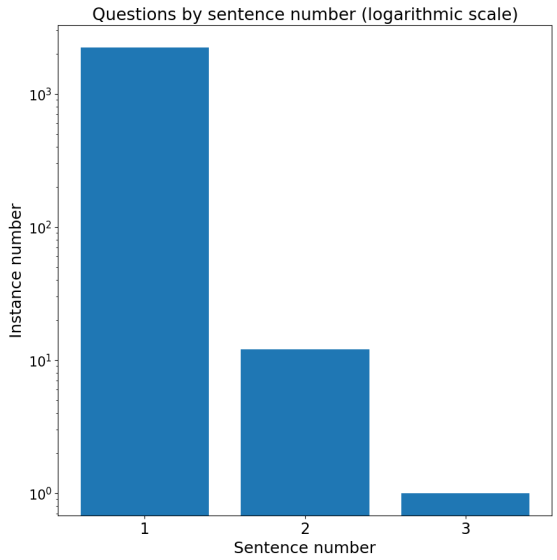

Figure 3: Distribution of the number of sentences among the BioASQ questions.

It is interesting to observe the distribution of the most frequent tokens among the different question classes as it may help to identify key tokens for each class, which could be a base for a simple classification system. In Figure 4, the 30 tokens with the highest number of occurrences in the dataset are shown, grouping their occurrences by question type. It should be noted that non-alphanumeric characters were not discarded as individual tokens, based on an initial hypothesis that some of them may be interesting to analyze. Unsurprisingly, that is not the case of the question mark, "?", that appears in relatively balanced way among all classes, since they are all questions; however, the dot, ".", is much more common in the list type. Furthermore, it is interesting to see how some tokens with the same characters but different case for the first letter (normally based on whether they are the first word of a sentence) have dramatically different

distributions among classes. For instance "Is" appears almost exclusively in binary (yes or no) questions, while "is" is mostly distributed among the summary and factoid questions. Therefore, one can infer that in the pre-processing of any model devised for solving the classification task, the case information can be useful, and should not be removed via global lower-casing of all tokens. Also, stemming does not seem a suitable option, since the class distribution of words with the same stem can significantly vary. This phenomenon is exemplified, for instance, in two tenses of the verb "be": "is" is mainly split between summary and binary questions, whilst "are" is only very common in list questions.

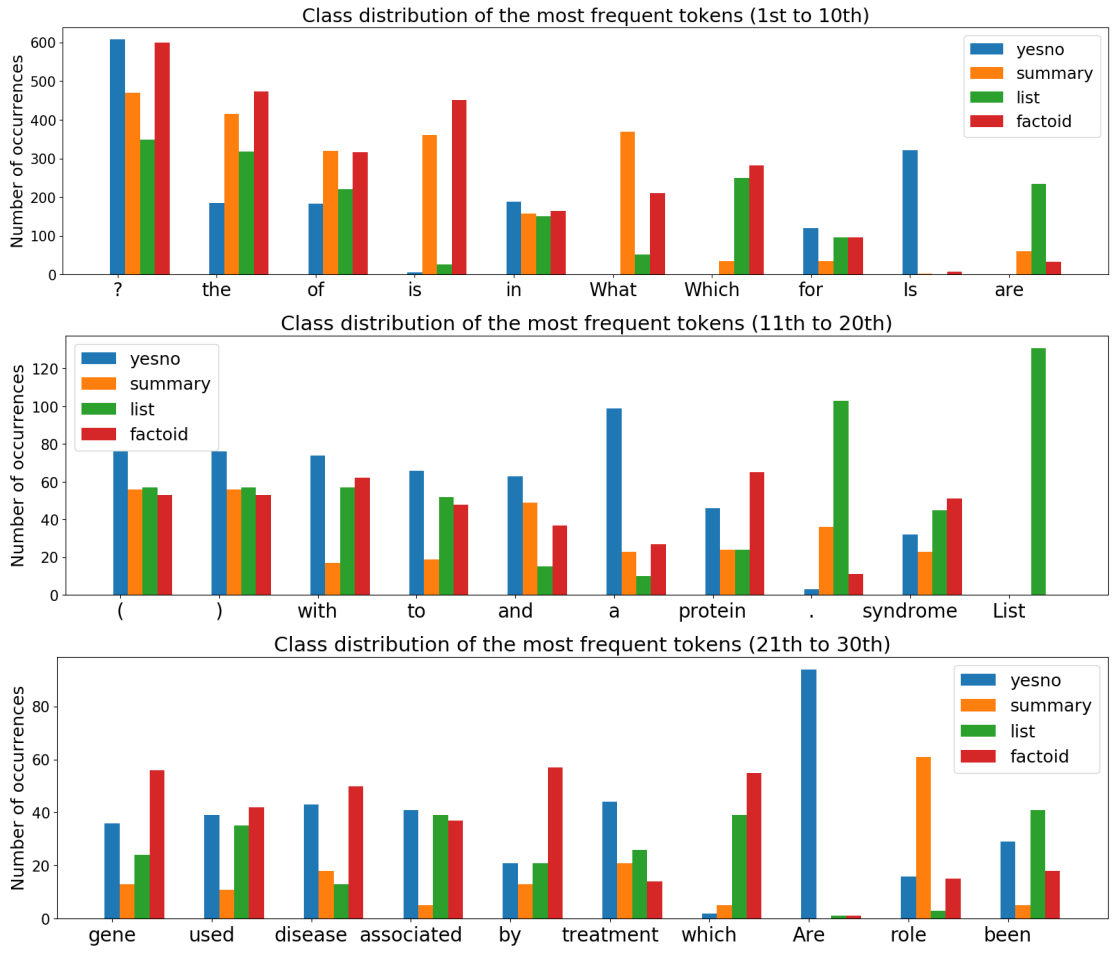

Figure 4: Distribution of the 30 most frequent tokens in question types.

3 Proposed Methods

3.1 Experimental configuration

Different models are proposed to solve Task 1 and Task 2, and they were trained and tested with the same data splits to compete in the same circumstances; since the BioASQ data is small, using different partitions can have a large influence on the results of a classifier. Before splitting, the data instances were shuffled, to avoid being affected by any hypothetical bias due to the original order of the data. Nevertheless, shuffling was performed in a deterministic way (using a fixed random seed) to preserve reproducibility and have a common setting for all models. 90% of the original BioASQ questions were used for training, and the remaining 10% for testing (i.e. more than 200 instances), which was considered enough to obtain meaningful results. Stratification was applied at splitting time to ensure that all classes keep (approximately) the same distribution in both splits (the same as in the whole data). The training set is used to generate the input for models to learn how to classify instances, while the test set allows to evaluate the classification on independent instances, as explained in Chapter 4.

The parameters of some models were fixed after trying different values. In such cases, to avoid an overspecialization on the test model that may lead to not generalizing well to other data, the model was trained with most of the training data, except for 11% (10% of the total data), which was used for validation; i.e. the value that yielded the highest validation F1-score was chosen as final for a given model. The validation split was not only used for parameter optimization but, most importantly, as an guidance indicator of the quality of the evolution of an incrementally trained model in Task 2, as explained in Section 3.5.

In all the models, the questions were tokenized to obtain their words. In some models the default tokenization provided by the NLTK package [3] was used alone, while in other models additional filters were considered, and they are commented in some of the following sections of this Chapter. Generally words were lower-cased in a pre-processing step (to be considered equal regardless of their case), except when otherwise stated. The tested Machine Learning models have certain non-deterministic steps, but random seeds were fixed to make them less variable in results in favour of reproducibility.

3.2 Baseline Rule-based model

In the Linguistic Analysis performed in Chapter 2, the 30 most frequent tokens among all the questions were shown, and it was obvious to see that some of them are significantly more common in one class (namely, at least approximately 2 times more frequent than the second one), which can help to define a rule-based model to classify a question based on its tokens, and use it as a baseline model to compare it with more complex approaches with Machine Learning algorithms. Using a simple baseline model is especially interesting to identify if a problem is simple or complex; if the baseline model was able to obtain accurate results, it may not be necessary to use Machine Learning techniques to solve the problem in a satisfactory way. However, using the whole data to infer the rules of

the rule-based model could be considered as a way cheating. Instead, the training and test partitions explained in Section 3.1 were using.

To design the rules, the 30 most frequent words among the questions of the training data were gathered, as well as their distribution through classes (analogously as done in the Linguistic Analysis, but now only for the training partition). For each word, the percentage of occurrence of each question type was found, as one can see in Figure 5.

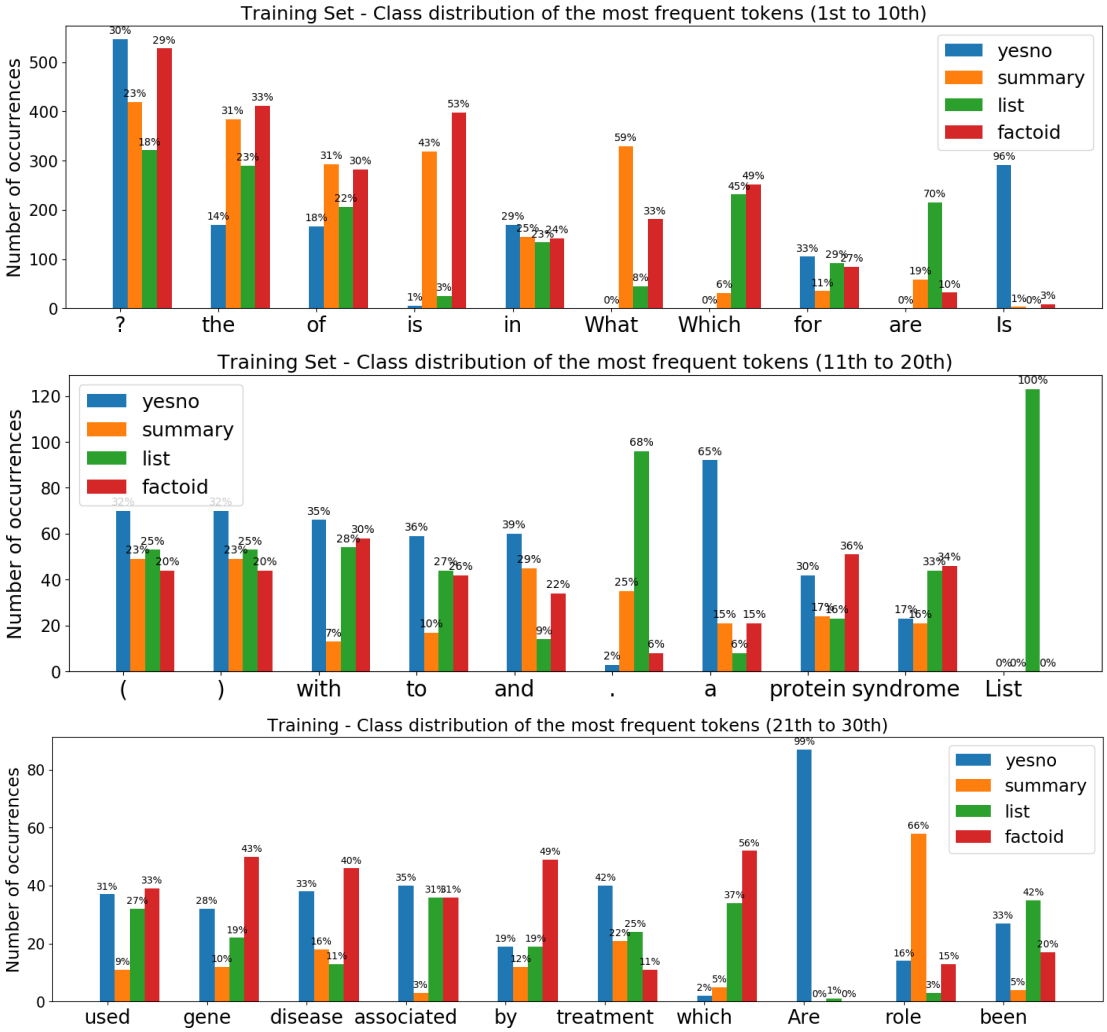

Figure 5: Distribution of the training set's 30 most frequent tokens in question types.

The Rule-based model was designed to work as follows. Each rule checks if the tokens of a (test) question contain a specific token that has been seen (in the training set) to appear much more frequently in one of the classes than in the others. In fact, an effective way to achieve a high accuracy is to sort the rules in descending order of confidence level,

where such level is considered to be the percentage of occurrences in the most frequent classes, that gets assigned by the rule to a tested question. Table 1 presents the rules in the order in which they are checked, showing the percentage of confidence extracted from the analysis depicted in Figure 5. As soon as one rule is validated, the associated type is assigned to the question, and subsequent rules are ignored. As a final (default) rule, if no other can be applied, the factoid class is assigned, since it is the most frequent among the training data (and the whole set of questions). All question types are represented in at least one rule, and they are limited to tokens with at least 50% of occurrences for a specific class, to have a simple set of rules, yet based on reasonable consensus from the training data. Note that the two rules before the last one can be removed since they do not have any effect on the results, since they assign the factoid class, just as the final default rule. Therefore, for simplicity, they were not used in the final Rule-based model.

Token	**Assigned type**	**Confidence (%)**
List	List	100
Are	Yes/No	99
Is	Yes/No	96
are	List	70
.	List	68
role	Summary	66
a	Yes/No	65
What	Summary	59
which	Factoid	56
is	Factoid	53
Default case	Factoid	–

Table 1: Rules of the Rule-based model, which assigns a class if a question contains a specific token.

3.3 Support Vector Machine model

The Linguistic Analysis of Chapter 2 demonstrated that some words can be clear indicators of the class of question that contains them, but it was desired to aggregate the information provided by multiple words in a more complex way than using simple rules (as done in Section 3.2), learning relations automatically, i.e. using a relatively simple and fast Machine Learning model. The Support Vector Machine (SVM) model was chosen, based on its success in text classification tasks in the literature [4].

An SVM model can use different features to encode words. The most simple way to approach this task might be to represent each question as a binary vector with the length of the vocabulary size, e.g. in a bag-of-words style. However, the high orthogonality of these representations (with questions having a value of 1 in the positions associated to the term they contain, and 0 in the rest) made this approach seem potentially ineffective. Instead, a floating-point vector representation was chosen, TF-IDF (term

frequency inverse document frequency) vectors, capable of weighting (approximately) the importance of words based on their frequency of occurrence in texts, i.e. common words with little meaning (stop-words, such as "a" or "the") are given much less importance than less common words, including most nouns, adjective and adverbs. Despite of the fact that the Linguistic Analysis showed that some of the words that can help determine the class of a question are common stop-words, it is considered that there might be a long list of less common, more meaningful words which can help to determine the class (such as "role", as seen in the Linguistic Analysis). It is well-known that models using TF-IDF vectors can be outperformed by more complex features (such as Latent Semantic Indexing [4]), but it was wanted to use quite simple features before trying the most complex yet promising ones, word embeddings, using neural models (in Section 3.4).

The SVM model was configured to use a linear kernel, after trying more complex alternatives (Radial-Basis and Gaussian) in the validation set and obtaining significantly lower results, probably caused because the high dimensionality of TF-IDF vectors make it easy to separate vectors without additional space transformations, and given the relatively low number of instances. The parameters of the SVM model include a cost value of 1, square-hinge loss and L2-penalty, which are the default in the scikit-learn library [5], since they usually work well in many different problems, even if not always [6], by balancing the class separability with the correct classification of instances.

3.4 Neural Network models

3.4.1 Motivation

The motivation of using a Neural model is based on the reported superiority of Neural Networks in text classification tasks in the literature [7], supported with our own experience in previous practical works pointing in that direction. We gained knowledge on how to design effective Neural models, which was useful to apply in this work. Since the Neural models were the main set of models chosen for this work, special attention was put in designing pre-processing mechanisms and features that seemed suitable to solve the classification task, as explained in the following sections.

3.4.2 Text pre-processing

Pre-processing is a key aspect of the pipeline of NLP tasks, which was not used (beyond tokenization and lower-casing) in the previous models (rule-based and SVM) due to the simplicity of their features.

In the Linguistic Analysis of Chapter 2, it was observed that the case of some words influenced the type of question to which they use to belong to (in the BioASQ dataset). It was seen that the interesting words whose first letter is capitalized occur mostly in the first word of the text, since most questions are made up of a single sentence (i.e. there are few dots in the middle of the analyzed texts). These findings motivated to define a pre-processing step that turns all words into lower-case, except for the first one in the text. However, an alternative strategy of lower-casing all words was also tested,

to study if it was more suitable, since it is traditionally used in many NLP tasks, to ensure that same words are considered equal independently of the case. This option has been motivated by the idea that the position of words (and therefore whether it is in the start of a sentence) can possibly be implicitly distinguished by a neural model, e.g. via convolutional filters or recurrent memory units.

In the tokenization step, multiple characters were filtered out (most ASCII non-alphanumeric characters), except dots (”.”), commas (”,”), parenthesis (”(” and ”)”) and question marks (”?”): these characters were considered important to delimit sentences, clauses and questions, adding extra information, closely related to the syntactic structure of the texts, even if not providing tokens that are semantically meaningful alone. Additionally, in a validation step, it was observed that keeping words with hyphens (”-”) as a single token (without splitting) slightly increased the accuracy of the model; a plausible explanation is that compound adjectives (e.g. ”well-known”) keep their original notation and are later identified with an independent meaning, rather than that of their component words individually. Moreover, it was chosen not to remove stop-words to avoid removing information, even if it is given by words with almost no individual meaning, but that helps to connect other words in a syntactically accurate way.

3.4.3 Features: definition and feature engineering

3.4.3.1 Word embeddings

In a text classification task, it is intuitive that words themselves can be useful to associate a text to a certain type, especially after seeing in the Linguistic Analysis (Chapter 2) the high correlation of the occurrences of some words with certain question classes for the BioASQ texts. The literature suggests that combining Neural Networks with a specific representation of words, word embeddings, can yield state-of-art results in text classification tasks [7], so it was decided to use them as features for the Neural Network models. Word embeddings are not only lexical features (related to words individually, and a whole vocabulary if studied globally), but also to semantics, since they define words using a low-dimensional numeric vector based on the neighbor words with which they appear in sentences, i.e. their local context. In other words, they are lexical-semantic features.

Since their inception, multiple word embedding models have appeared. There are many possible configurations, and some work better for some tasks than others, so it was chosen to opt for different alternatives and evaluate them, to see which one was the most suitable choice for the BioASQ challenge. The first selected type of word embeddings were trained specifically in the data of the task to solve, i.e. focused solely in that vocabulary, which is smaller than in a general domain. For this purpose, Word2Vec embeddings [8] were trained using Python’s Gensim package [9], using a skip-gram model with a minimum word count of 1: since the corpus is small, it was expected that many words would only appear once, yet still be relevant. 128 dimensions was considered a suitable size for the vectors, given small vocabulary of the data: after pre-processing there are 4444 unique tokens.

GloVe embeddings constitute a well-known model of embeddings built using unsupervised learning, associating the meaning of words based on their co-occurrences [10]: they are the second type of embeddings that was tested. In this case, the embeddings were not trained from scratch, since they had been pre-trained in a very large (27-billion-token) Twitter corpus. Using these 200-dimensional embeddings was considered interesting to see the consequences of applying broad-domain embeddings in the narrow-domain of bio-medicine to which the BioASQ questions belong. While it would be possible to fine-tune the GloVe embeddings to try to adapt them to the biomedical vocabulary, the broadness of the original domain in which those embeddings were trained may still make it difficult to obtain significant improvements or changes in the word representations.

The fine-tuning approach was instead tested in a third representation type, the BioWordVec embeddings (presented recently, in 2019), reported to be capable of obtaining better word similarity results in biomedical corpora than former state-of-art methods [11]. As their name implies, these embeddings had been built using biomedical corpora, which matches the domain of the BioASQ questions and provides an interesting advantage with respect to other options, since even very specific words of the biomedical domain can potentially have solid embeddings learned on a large-scale corpora (BioWordVec represents more than 2 million words). More specifically, the "extrinsic" version of the embeddings were used, since it is mentioned by the authors to be suitable to apply it to NLP tasks in the biomedical domain. The embeddings have 200 dimensions and were learnt using a window size of 5, which is small enough to adapt well to many domains [11]. As mentioned, the pre-trained BioWordVec embeddings were re-trained on the BioASQ data to specialize them on its own sub-domain. A Word2Vec model was used to perform the fine-tuning step.

3.4.3.2 POS-tag embeddings

Although word embeddings are common features in the state-of-the-art of NLP tasks, the BioASQ challenge involves identifying the class of considerably short questions (as seen in the text-length study performed in Chapter 2), which leads to think that some other features might be effective too. Namely, it was considered that studying the morphology of the words might be useful too, even if it implies a simplification of the meaning, e.g. the "VBP" Part-Of-Speech (POS) tag associated to a verb is more ambiguous that fully specifying verbs, such as"influence", "affect" or "enumerate". Of course, the order of POS tags (knowing that one tag is obtained for each word) is important, learning the correlation of small (local) sequences of POS tags might be helpful to classify the questions. To obtain this kind of representations, POS-tag embeddings [12] were built using Word2Vec in an analogous way to the one used to obtain word embeddings explained in the previous section. The POS tags of the words of each text were obtained using NLTK's tagger [3], obtaining tags in the Penn-Treebank notation.

3.4.3.3 Syntactic dependency embeddings

Following a similar reasoning as in the previous section, syntactic embeddings were chosen to learn local adjacencies or sequences of syntactic dependency relations, that can be key to classify a question, e.g. it was observed in Chapter 2 that some question types (e.g. list) use indicative sentences instead of interrogative, and it is known that sentence modality can be inferred from analyzing the syntactic relations of words. Namely, syntactic triples are used, where each one is defined by three elements: *(head word's POS tag, syntactic relation, modifier word's POS tag)*. It should be noted that it was wanted to generalize triples from words to a morphological-syntactic level, by not storing the specific words but their POS tags; otherwise, for a small corpus as the BioASQ one, it is likely that a large number of triples would never occur more than once, making it difficult to learn embeddings (learning their context throughout multiple sentences). It was considered that 128 dimensions would be enough to represent triples in a satisfactory way, since the number of POS tags and syntactic relations is by no means massive. As it happened with POS embeddings, syntactic embeddings are not new in the literature [12], despite of being much less common than word embeddings. To obtain the triples, Stanford University's CoreNLP [13] dependency parser was used.

3.4.3.4 Feature aggregation

In the previous sections, multiple features based on embeddings were introduced, namely word, POS-tag and syntactic-triple embeddings. Whether it is possible to use each of them individually, it is also valid to combine the features when it comes to train the classifier. To assess the advantages and drawbacks of these options, multiple feature combinations were tested, including each of the models alone (with the three types word embeddings used exclusively or aggregated), or joined together. In the cases in which multiple features were aggregated, they were preserved in their original formulation for simplicity (not condensed in a complex combination): the features are vectors that can be concatenated for each text, regardless of the dimensionality of each vector. A pipeline of the whole classifier system is depicted in Figure 6, which presents a visual overview of the possible features of the Neural Network model, that can be either aggregated or excluded.

3.4.4 Neural Architectures

In the state-of-art of Neural Networks used in text classification, three of the most common architectures are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and hybrid models of the previous two [7]: the three options were tested in this work (built using the Keras library [14]). CNNs were considered interesting due to their ability to identify local patterns that highly correlate with certain output classes: in this case, local combinations of words at certain positions (or depths) of the textual structure, using 1-D convolutions. RNNs seemed promising due to their ability to deal with sequential data (here sequences of words), leveraging the information extracted

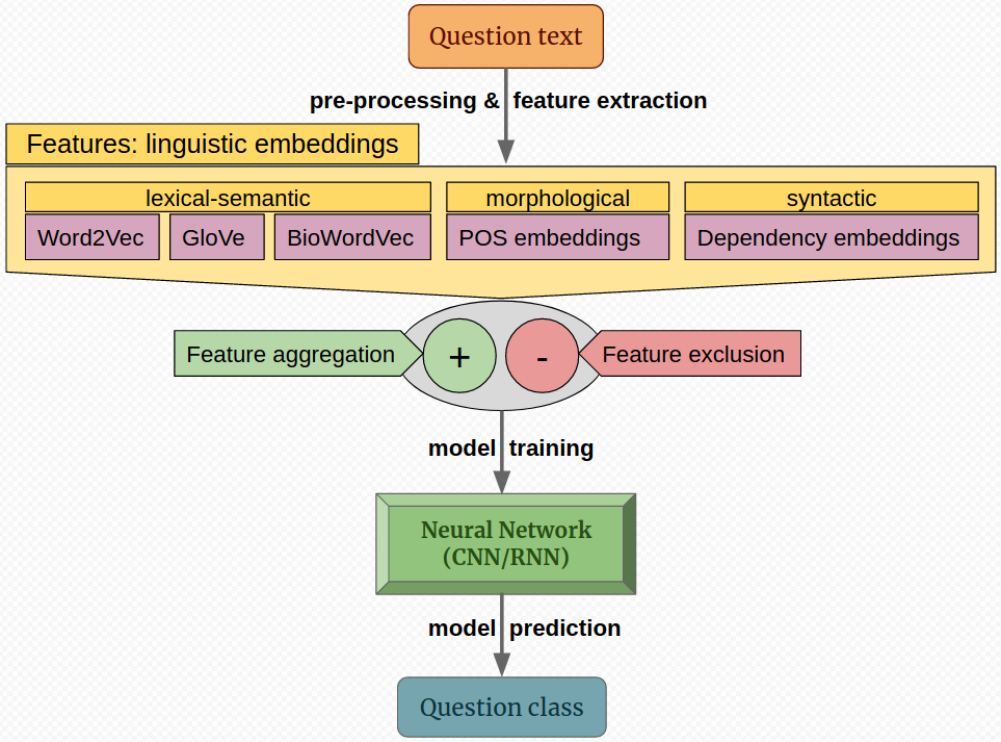

Figure 6: Pipeline of the Neural Network classifiers.

from the first elements (and kept memory units) with most recent (late) data. Intuitively, combining both architectures may opt to obtain the advantages of convolutions and recurrence, as long as the greater complexity does not cause overfitting. Precisely to avoid overfitting, all the designed architectures use a rather small (not very deep) number of layers and a moderate number of hidden units, combined with dropout layers to preserve generalization, and prevent overspecialization in the training data. Interestingly, even if there are not many instances, having a high dropout (e.g. 0.3 or 0.5) was counter-productive in terms of validation accuracy, probably due to the compact, not very deep architecture, for which a dropout rate of 0.15 gave the best validation results.

Since all architectures solve the same problem using the same set of features, they have the same input layer, which is an embedding layer that receives a matrix of embeddings, that contains the word embeddings, POS embeddings or syntactic embeddings of the questions, or an aggregation of them. All the architectures end with a softmax layer with 4 units, one for each of the possible classes that can be predicted, and the one with the highest activation value is selected. The intermediate layers are different for each model. The CNN has a single convolution layer of 300 filters of mask size 5; using more filters or higher mask sizes gave less validation accuracy, probably due to the low size of the BioASQ data (causing overfitting), while a considerably smaller number of filters (or using very small window sizes) was not enough to learn appropriately (underfitting). The

CNN model uses global max-pooling after the convolution layer to reduce the complexity of the model by keeping the most significant higher activations of the convolutions. Before convolution and after pooling, dropout layers are used to promote generalization, with the last one followed with a fully-connected (dense) layer to increase the abstraction capability of the model (with 500 hidden units, that provided better results in the validation set than more complex models, that reached overfitting, or simpler configurations, being prone to underfitting), before the softmax layer. The described architecture of the CNN model is depicted in Figure 7.

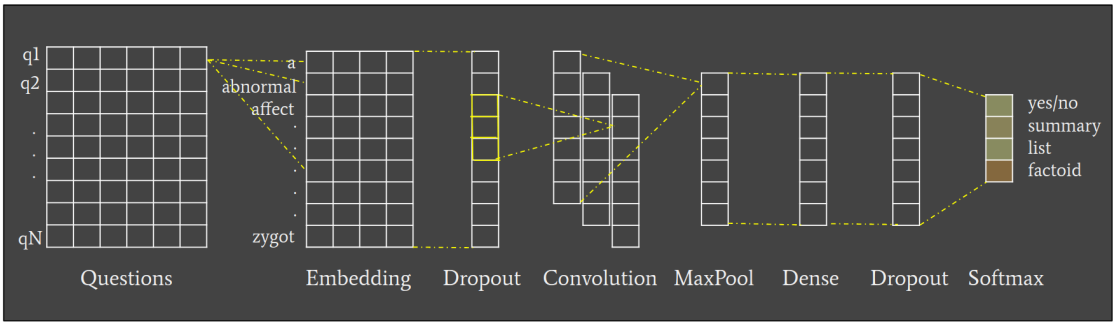

Figure 7: Architecture of the Convolutional Neural Network model.

In the state-of-the-art of RNNs, both Long-Short Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs) are commonly used, also in NLP [15]. Therefore, a GRU model and a LSTM architecture were tested, using very simple configurations, with only 32 and 64 recurrent units (respectively), selected using the validation set, that also lead to discard adding additional fully-connected layers, that caused overfitting; the recurrent units are enough to extract abstract representations of the data to classify. The hybrid CNN-RNN architectures uses the same structure as the CNN, but placing a GRU or LSTM layer just after the initial embedding layer (with the same units as specified in this paragraph).

3.5 Incremental Co-traning with external Quora questions

3.5.1 Biomedical and consensus filtering of questions

The Quora dataset contains 404,302 question pairs, i.e. 808,604 questions in total, and they are of a very broad-domain, while the objective of this work is to maximize the predictive power of classifying BioASK questions of the biomedical domain. Therefore, it was considered appropriate to filter out questions of other domains, and only use those identified as biomedical. While it would be possible to build an entire binary classifier to determine if a question is biomedical, a much simpler and faster decision system was used. A comprehensive list of almost 100,000 medical terms was used for this purpose [11]. The main idea was to check each (tokenized) word of every Quora question, and compute the proportion of words present in the biomedical vocabulary: texts with a high proportion were considered biomedical, while the rest were discarded. Common

words (mainly stop-words) were ignored in the calculation, since they are expected to appear in any kind of text. It can be argued that some biomedical questions can use very few biomedical terms and therefore they would not be returned by this system, and it is true. However, it was observed that relaxing the threshold lead to the inclusion of many non-biomedical questions, e.g. "You give me headache" would obtain a high proportion of medical terms, since "headache" represents the 25% of the words, or 50% if "you" and "me" are considered stop-words and ignored. Therefore, the threshold was increased up to the point (38% of bio-medical words, excluding stop-words) in which it was observed that most filtered texts were genuinely related to the biomedical domain, but not being too strict to eliminate all outliers, to conserve a reasonable number of Quora questions. Since Quora questions come in pairs of very similar questions, it was imposed the additional restriction that the two questions of a pair should be considered biomedical for any of them (actually both) to be kept. The final number of question pairs considered by the system was 8956, i.e. a bit more than 2% of the original questions is considered to be biomedical.

After the biomedical filtering, another criterion is used to give greater consistency to the model: only the question pairs with the same class prediction by the initial model are used. After observing part of the biomedical-filtered Quora data, it was seen that many pairs (apparently about half of them, or even more) have the same question type, and since the true labels of questions are not known, it can be assumed that if two pairs have the same class (knowing that they normally have the same type in reality), it is more likely that the class assignment is correct, even though there is no strict guarantee. Nevertheless, it is known that there are many exceptions to the class match of pairs, and those were automatically discarded as a side effect of this practice, e.g. "What's the best digestive enzymes?" should be considered as a summary-requesting question, while its partner question "Do digestive enzymes help?" is a binary yes or no question. Unsurprisingly, since the original models were trained with BioASQ questions, they failed to classify a significant number of Quora instances, even if many of them effectively belonged to the biomedical domain; maybe the more informal language used in Quora (e.g. using contractions) and other linguistic aspects were not known in the original BioASQ dataset, and an important fraction of the vocabulary is new (other than common words and some biomedical terms). Additionally, questions classified as belonging to the binary (yes or no) type were ignored, since the models of Task 1 had already learned to detect it very well (with a perfect F1-score of 1 for some CNN model, e.g. the one with BioWordVec embeddings).

In the end, the number of questions of the pair-wise class-match consensus criterion (after applying the biomedical filtering) depends on the specific model, e.g. 1411 question pairs (2822 questions) for an execution of the CNN model with BioWordVec features. Other runs of this model and different ones gave a different number of questions, but in no case more than 3800 questions (and never less than 1700 question pairs). In any case, the number of obtained questions is a bit smaller or greater than the size of the original BioASQ questions, so its size is significant enough to potentially change the predictive capabilities. The percentage of Quora questions that were classified with the same class

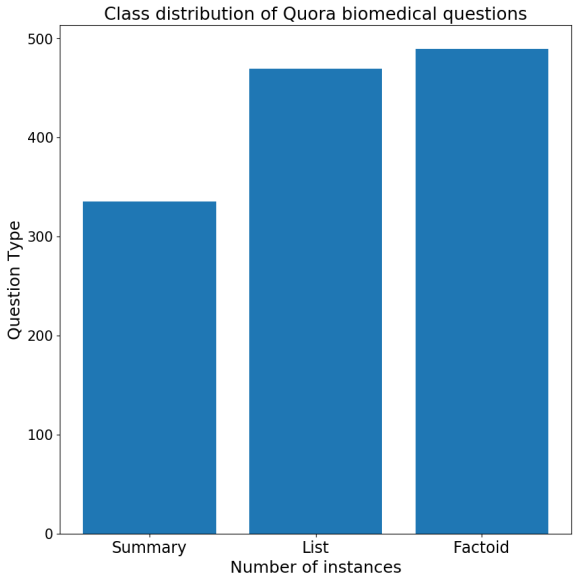

Figure 8: Predicted class distribution of filtered Quora question pairs.

ranging from 32% to 46%. As mentioned, some of the non-matched cases were caused by original pairs belonging to different question types, while the rest were classification errors of the initial model. The percentage of correctly classified Quora question pairs was observed to increase during the execution of co-training, being able to exceed 50% after few iterations, exceptionally surpassing 65% in some cases. An example of the predicted class distribution of filtered Quora pairs is shown in Figure 8; as mentioned, binary questions were ignored due to the very good performance of models just using the BioASQ data. As it happened with the BioASQ questions, factoid is the most commonly assigned type, but in the Quora questions there are more instances predicted as list, while many less summary ones. However, this might not reflect reality in a faithful way, since a brief review of the labels in 20 Quora questions showed 6 errors, all of them confusing summary with factoid or vice versa; it cannot be discarded that there is a higher true proportion of summary questions, some of which were mistakenly classified as factoid.

3.5.2 Co-training cycle

As mentioned in Section 3.1, the co-training scheme of Task 2 uses 80% of the original BioASQ questions for training (at the initial step), 10% for the validation (the same for every step), and 10% for the final testing of the model. While the test set is the same as in Task 1, the final models used in that task used a higher percentage of the original BioASQ questions for training (90%), which intuitively gave them an initial advantage. However, the co-training strategy used in Task 2 subsequently adds more data in a series of iterations, in which new questions from the broad-domain Quora dataset (selected using the filtering criteria explained in Section 3.5.1) are incorporated into the

training set, labelled with the class assigned by the classifier of Task 1. The filtered Quora questions are divided in groups of the same size batches. After testing different values in the validation set, it was found that 10 iterations was the most suitable number; using many more implied adding very few data each time and training the model again, which was not very practical time-wise, whilst having too few iterations lead to adding many questions at once, i.e. merging in the same groups questions that might or might not be appropriate for the task, decreasing the filtering capability of the model, compared to using more batches with fewer questions.

In each iteration, the questions of one batch are added to the training set, and the model is trained again, to incorporate the knowledge extracted from the new questions. Then, the model is used to predict the classes of the questions in the validation set, formed entirely by original BioASQ questions (which are not part of the training or test partitions, of course), since the ultimate goal of Task 2 is still to maximize the predictive power in the BioASQ data. Therefore, if the validation accuracy obtained in one iteration is higher than in the previous one, it is assumed that the test accuracy would also be higher (since both splits are formed of BioASQ questions with equivalent class distribution), and hence the most recently introduced batch of Quora questions is accepted: the knowledge extracted from them has been useful to improve the predictive power in the BioASQ context. Otherwise, if the validation accuracy is lesser than in the previous iteration, the new Quora questions as discarded, since they introduced noise that blurred the classification ability in the BioASQ corpus; the model is restored to its previous state. After all the iterations have been performed, the remaining model is the one trained with the combination of BioASQ and Quora questions that yielded the highest validation accuracy, and is then used to predict the class of the test questions, and assess the quality of the model. An overview of this co-training cycle is depicted in Figure 9.

The results presented in Chapter 4 explore whether the usage of more data (the Quora questions) brought an eventual improvement of the classification capacity, for different Neural Network models which performed in a satisfactory way in Task 1, outperforming other models (including the Rule-based and SVM classifiers) whose results were not so good (as examined in Chapter 4).

3.5.3 Discarded Co-training strategies

This section covers two slight variations of the co-training cycle explained in Section 3.5.2. These alternative schemes were discarded since they caused a decrease (or simply no improvement) in the validation accuracy of the tested models, which were not all the Neural Network architectures, but at least the CNN model with Word2Vec or BioWordVec embeddings. Therefore, they are not included in the final results of Chapter 4, that focuses on the test results (not validation ones) on a wide range of models, since it was considered more interesting to analyze how the best co-training scheme affected different feature configurations.

The first alternative co-training scheme did not predict the class of Quora at start like the final model, but at every iteration, just before incorporating those specific

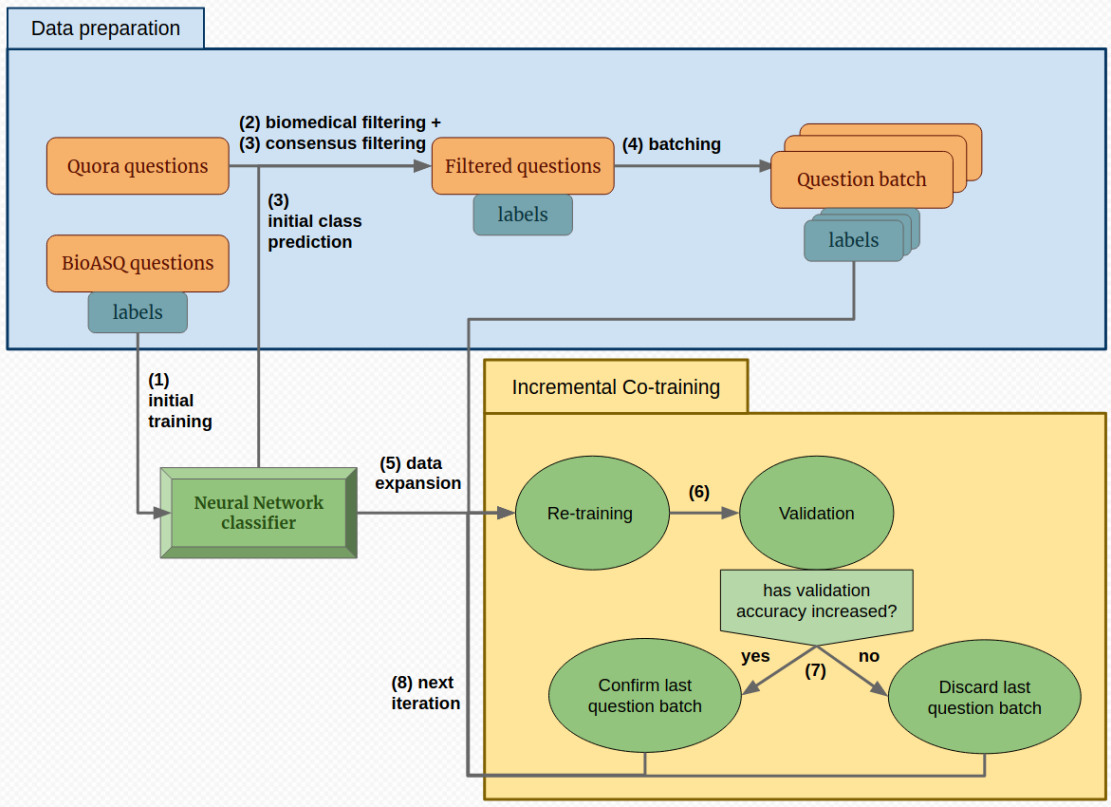

Figure 9: Co-training cycle used in Task 2.

questions into the training set. This was based on the idea that, if a model is able to be refined throughout iterations, its predictive power might increase, and eventually predict better the class of the questions of the last iterations (better than initial predictions). However, such configuration yielded a F1-score of the same significance and quality as the other approach for few models, maybe due to a propagation of the error introduced by mislabeled instances among Quora questions (not manually annotated, but using a classifier), in contrast with fully-reliable annotation of BioASQ questions (even if they are less). Therefore the incremental prediction approach was discarded, since predicting all classes at the beginning was a slightly more simple process, since the filtering related to matching of question pairs (explained in Section 3.5.1) were performed just once, and the final size of the Quora questions was then known from the very start.

Another interesting scheme that was considered as an evaluation measure at every incremental step (instead of the accuracy in BioASQ questions of the validation set), was to check the percentage of Quora question pairs labelled as duplicate that were assigned the same class by the classifier. This was motivated by the fact that duplicate questions are paraphrases, so they can be considered that, without any doubt (as ground truth), they belong to the same question type. Therefore, if the classifier improves the percentage of matched types among duplicate question pairs from one iteration to another, one may

think that it is likely that the predictive power has increased. However, in this way, the classification is being validated on questions of a different domain (the broad-domain of Quora) instead of those the domain of the target (BioASQ) questions, and the acceptance or discarding of question batches is then not specialized in adapting to the target domain. Therefore, it is unsurprising that this approach provided a slightly lower F1-score when tested in a specific Neural Network model, that lead to discarding it in the final model, that preserved validation accuracy.

4 Results and Discussion

4.1 Evaluation metrics

The results presented in this Chapter depict three different metrics: precision, recall, and F1-score. The precision is interesting because it compares the number of true positives by such number plus the false positives, i.e. a low precision is caused by having many false positives. Since recall is the true positive rate, it is useful to see how many instances of a class (or overall) are correctly labeled by the classifier; a low recall implies having many false negatives. The F1-score is the harmonic mean of precision and recall, and a common measure in scientific papers to report final results. The three metrics are shown for some models in a per-class basis, in cases in which it is interesting to analyze the predictive power for specific classes, to try to understand how a classifier works, i.e. its strengths and weaknesses. For these models and many others that were also tested but provided less relevant results, an overall measure of the precision, recall and F1-score is shown. For the overall results, weighted measures are used, i.e. those resulting of being the mean of all classes, but weighting each class proportionally to the number of true instances it has in the test data, which is more realistic when classes are not perfectly balanced. However, it must be said that other popular versions of the metrics, such as micro- and macro-average, obtained very similar results, with differences below 2%, normally closer to 0% or 1%.

4.2 Task 1

4.2.1 Rule-based model

The results of the rule-based model, presented in Table 2, are very different from one class to another, with very different metrics for a single class, and it is worth analysing this phenomena. The binary yes/no class is the easiest to classify (not only by this classifier but also by the models explained in the next sections), due to its characteristic starting words, such as "Is" and "Are", which are among the first rules checked by the rule-based classifier, due to their clear distribution in favour of the yes/no class, with confidence percentages of 96% and 99%, respectively, which explains the high precision (not many false positives). However, there are more false negatives (lower recall), since some other words that are typical in yes/no questions are not considered, e.g. "Does" (since it was not among the most common words in the BioASQ dataset). The high number of false

positives in the summary class can be explained by the lower confidence (not-so-favorable distribution) of the words used in their rules: "role" (66%) and "What" (59%), which is also quite common (33%) in factoid questions, leading to some false negatives (mediocre recall) for such class, that also has many false positives (low precision) due to the low confidence (and therefore low priority, i.e. late checking) of their rules, "which" and "is", somewhat compensated by the default rule that assigns the factoid class, because it was the most numerous one in the training data, as examined in the Linguistic Analysis of Chapter 2. The worst predictive power is that of the list class, despite of having the most certain rule, saying that a question with the word "List" is of the list type with 100% support in the training data, which yields a decent precision, but cannot prevent obtaining a very poor recall. A plausible explanation is that many list questions do not contain the "List" term, nor the other two terms that have rules for this type, therefore causing a high number of false negatives.

Class	**Precision**	**Recall**	**F1-score**
Yes/No	0.83	0.61	0.70
Summary	0.51	0.73	0.60
List	0.67	0.22	0.33
Factoid	0.41	0.52	0.46
Weighted avg	0.59	0.54	0.54

Table 2: Evaluation results of the rule-based classifier.

The overall results of the rule-based model are mediocre, with an F1-score of 0.54, being more handicapped by the false negatives than false positives, i.e. the recall is clearly lower than the precision (0.54 compared with 0.59). Adding more rules before the default factoid-class assignment may increase the recall of all classes (except factoid, unless it was very well represented in new rules). However, the main drawback of the rule-based model is that it only considers a single word per rule to classify a whole text, which is overly simplistic in non-trivial domains with sentences that are not extremely short on average. Hence, a rule aggregation strategy, i.e. combining the activation of multiple words with different weights (proportional to their confidence level), analogously to a Fuzzy Rule-based system, would probably increase the predictive power of the model. However, this model was conceived to be as simple as it is willingly, to see how far one could reach in this task with simple rules. The results are obviously better than a random classifier that, for a more-or-less balanced problem with 4 classes would get a lower accuracy of roughly 25%. The models analyzed in the next sections aggregate features extracted from words in more advanced ways (using Machine Learning) that clearly outperform the rule-based model.

4.2.2 Support Vector Machine model

The test results of the Support Vector Machine model, shown in Table 3, are much more stable among classes and metrics. Namely, the class with the lowest F1-score (0.54),

summary, is not very far away from the one with the highest result, list (0.67), yielding an overall weighted F1-score of 0.61, which coincides with the precision and the recall, and is a clear (but not massive) improvement from the rule-based model, that had a F1-score of 0.54. The additional predictive power obtained by the SVM can be explained by its extraction of TF-IDF vectors describing the questions based on their words (and, more specifically, their frequencies among the training data). The model learned patterns of words (represented as frequencies) that when occurring together were highly correlated with a question type (separable in a common region of the SVM's hyper-space). However, this method has some disadvantages, especially seen for words that appear very few times (especially if just once), since the model is prone to associate them with the class of the few instances that contained them. The Neural Network models explained in the next sections add more complexity to the aggregation of word-derived features, in an attempt to tackle the limitations of the Support Vector Machine model.

Class	**Precision**	**Recall**	**F1-score**
Yes/No	0.62	0.69	0.66
Summary	0.54	0.54	0.54
List	0.69	0.65	0.67
Factoid	0.60	0.56	0.58
Weighted avg	0.61	0.61	0.61

Table 3: Evaluation results of the Support Vector Machine classifier.

4.2.3 Comparison of Neural Network models with different features

All the tested Neural Network models obtain results of much higher quality than both the rule-based and SVM models, as shown in Table 4. In all these cases, the Neural Network models use the same architecture (CNN), which yielded the best results (the other architectures are analyzed later in this section). Also, the CNN models of the table use the pre-processing variant that lower-cases all the tokens, since it was more effective than alternative configurations (explored later in further paragraphs). Curiously, the usage of Word2Vec embeddings trained specifically and exclusively on the training BioASQ questions was slightly less effective (with an F1-score of 0.83) than using broad-domain GloVe embeddings (with an F1-score of 0.86), that may not contain some of the technical biomedical words that appear in the BioASQ data, but compensates this fact by providing a more solid foundation in the embedding construction process for more common words, since they are based on a large number of examples, that help to get a more solid semantic representations of them. Unsurprisingly, the usage of the domain-specific (biomedical) BioWordVec embeddings (that are then fine-tuned for greater adaptation to the BioASQ questions) is even more effective, obtaining a F1-score of 0.89. In this case, the original embeddings were trained on a large-enough to learn solid representations for common words, and much better than the other embedding types for the case of biomedical vocabulary, due to the domain of the original corpora. The concatenation of multiple word embeddings is slightly less effective than using a single

type, even just Word2Vec (the F1-score for aggregated approach range from 0.83 to 0.81, where the worst option is to combine the three embedding types). This circumstance can be explained by the differences in the original corpora and architecture of the three embedding models, that lead to represent the same words in very different vectors that are not easy to (meaningfully) assimilate or leverage, even for a Neural Network. Some kind of translation or common-space mapping may be required to align these representations and let they enrich the predictive power.

In the case of non-word embeddings, one can see that both POS-tag embeddings and syntactic-dependency embeddings obtain a F1-score of 0.80, which is not as good as that of using word embeddings, but not very far, and still much better than SVM. This can be explained that extracting and aggregating local patterns of POS tags or syntactic dependencies in questions is still helpful to determine their type, but the variety or level of specificity that they can provide is lesser than the words themselves, as using in word embeddings. The simpler vocabulary of non-word embeddings may be interesting to apply to other Machine Learning models, including SVM, but a deep-enough Neural Network model can take the full potential of using embeddings directly derived from a vocabulary of words, even if it is considerably high. It is not very surprising that the results of using syntactic dependencies and POS tags is so similar, since the notation used for dependencies (synactic triples) actually includes POS tags as well. As it happened with word embeddings alone, the combination of multiple types of embeddings does not provide extraordinary results, but combining BioWordVec with POS-tag embeddings or syntactic embeddings yields better results that only using non-word embeddings. This can be explained by the simpler nature of non-word embeddings (due to their smaller vocabulary), that can facilitate the implicit task of the Neural Network to aggregate the meanings of multiple embeddings. Another reason to consider is that BioWordVec embeddings provide alone quite better results than non-word embeddings, so it is easier to improve the model by adding them.

The results of the best CNN model (with BioWordVec embeddings as features), with a weighted F1-score of 0.89, are shown in Table 5. It is easy to see how the model learns the core word patterns of yes/no questions perfectly, obtaining a F1-score of 1. The other question types are more difficult to identify, and, based on an observation of the data, the most difficult task (also for a human) is to distinguish between the summary and factoid questions, that sometimes use similar words and have only subtle differences (their F1-scores are 0.85 and 0.89, respectively), while the list type is a bit easier to detect (less difficult to cause confusion), obtaining a F1-score of 0.89.

As mentioned in Section 3.4, an alternative, less conventional pre-processing strategy was also tested for Neural Network models, consisting in not lower-casing the first word of a question, based on the clearly different question type distribution of some questions based on whether they appear at the beginning of a sentence or not (having different case), as observed in the Linguistic Analysis of Chapter 2. However, as one can see in Table 6, such approach was less effective (0.83 of F1-score) than lower-casing all words (0.89), using a CNN model with BioWordVec embeddings in both cases. A plausible explanation of this circumstance is that a CNN applies filters throughout the length of

Model	Weighted Precision	Weighted Recall	Weighted F1-score
Rule-Based	0.59	0.54	0.54
SVM	0.61	0.61	0.61
CNN (Word2Vec embeddings)	0.83	0.83	0.83
CNN (GloVe embeddings)	0.87	0.87	0.87
CNN (BioWordVec embeddings)	**0.89**	**0.89**	**0.89**
CNN (Word2Vec+BioWordVec embeddings)	0.86	0.86	0.86
CNN (Word2Vec+GloVe embeddings)	0.83	0.82	0.82
CNN (GloVe+BioWordVec embeddings)	0.82	0.82	0.82
CNN (Word2Vec+BioWordVec+GloVe embeddings)	0.81	0.80	0.80
CNN (POS embeddings)	0.81	0.80	0.80
CNN (Syntactic embeddings)	0.80	0.80	0.80
CNN (BioWordVec+POS embeddings)	0.80	0.80	0.80
CNN (BioWordVec+Syntactic embeddings)	0.82	0.81	0.81
CNN (BioWordVec+POS+Syntactic embeddings)	0.83	0.83	0.83

Table 4: Results obtained by different models with the specified features.

the sentence, and it is easy for the model to implicitly learn the implications of having certain words in specific positions of a sentence (e.g. at start), without the explicit need of calculating two different embeddings, one for each case. Actually, the embedding duality is counter-productive for learning the semantics of a word, since it implies having two separate representations for the same entity, that can lead to more meaningful results if combined together.

In Table 6, it is also possible to see the results of using alternative Neural Network architectures, that introduce recurrent units. It can be observed that none of these models are as effective as CNNs, probably because the questions are quite short, not

Class	Precision	Recall	F1-score
Yes/No	1.0	1.0	1.0
Summary	0.87	0.84	0.85
List	0.86	0.93	0.89
Factoid	0.84	0.81	0.83
Weighted avg	0.89	0.89	0.8

Table 5: Results of the CNN model with BioWordVec embeddings, in Task 1.

long texts where having memory of the sequential data (words) might be critical (and tricky, since some kind of implicit or explicit coreference resolution might be needed). Instead, CNNs are excellent for detecting small local patterns of words that correlate with question types, which is enough to outperform recurrent methods in this classification task. Interestingly, LSTMs have greater predictive power (0.83 F1-score) than GRUs (0.80 F1-score), maybe due to their slightly more complex architecture (using more units), but this complexity may become too much and lead to overfitting if combined with convolutions (0.79 F1-score), while combining GRUs with convolutions brings better classification capabilities (0.84 F1-score).

Model	Weighted Precision	Weighted Recall	Weighted F1-score
CNN (BioWordVec embeddings)	**0.89**	**0.89**	**0.89**
CNN (BioWordVec embeddings, first word not lower-cased)	0.83	0.83	0.83
GRU (BioWordVec embeddings)	0.81	0.80	0.80
LSTM (BioWordVec embeddings)	0.83	0.83	0.83
CNN+GRU (BioWordVec embeddings)	0.85	0.84	0.84
CNN+LSTM (BioWordVec embeddings)	0.80	0.79	0.79

Table 6: Results obtained by different Nueral models with the specified features.

4.3 Task 2

Table 7 presents the results of evaluating different models with the co-training scheme that introduces the usage of Quora questions to try to improve the classifications. As one can see, the predictive power is indeed slightly improved in some models, namely those that use word embeddings, but kept equal for those using non-word embeddings. As one can see, models with multiple types of embeddings or alternative architectures (other than CNN) were not tested in Task 2, since they proved to be less effective in Task 1. Even if non-word embeddings also yielded worse results than word embeddings, they were different enough from word embeddings to add interest to additional analysis of the impact of adding Quora questions in Task 2. The model with the highest improvement from Task 1 is the CNN model with Word2Vec features, with 2.41% of increase. This circumstance makes sense, since these embeddings were the only ones that had been originally trained in a very small set of texts (the BioASQ questions), not in a large corpora (as BioWordVec, where the improvement was only of 1.13%), or massive one (as GloVe, with 1.15% of improvement). Using Quora questions helped to refine the models powered with word embeddings, since more examples were added to define the meaning representation of word vectors.

The incorporation of more training data to a classifier can lead to improve its predictive

Model	Weighted Precision	Weighted Recall	Weighted F1-score	Improv. percent. from Task1
CNN (Word2Vec embeddings)	0.85	0.85	0.85	**+2.41**%
CNN (GloVe embeddings)	0.88	0.88	0.88	+1.15%
CNN (BioWordVec embeddings)	**0.90**	**0.90**	**0.90**	+1.13%
CNN (POS embeddings)	0.80	0.80	0.80	+0%
CNN (Syntactic embeddings)	0.80	0.80	0.80	+0%

Table 7: Results obtained by the different models used in Task 2.

power, under some circumstances. Firstly, the new data should be reasonably related to the test data, which was achieved in this case by filtering out non-biomedical Quora questions, as explained in Section 3.5.1. Another key condition is that the class labels of the new examples are correct, which cannot be guaranteed in this case, since the new data is not manually annotated, but labeled by the initial classifier. This circumstance can cause the propagation of error, which is obvious in many confused predictions of summary and factoid, as explained in Section 3.5.1. This lack of high quality (automated) annotation of new instances is detrimental in the learning process, and can jeopardize the success of the classifier if not managed carefully. To reduce the negative impact of this reality, the validation accuracy checks were introduced, to discard batches of Quora questions that deviated too much from the BioASQ data, bringing a negative contribution to the model. Thanks to this mechanism, the model can still find some usefulness in some Quora questions, and eventually improve the classification power when using word embeddings. To have a more detailed understanding of the co-training process, it is interesting to examine the evolution of the validation accuracy of the model, as shown in Figure 10 for the Word2Vec-powered CNN model, and in Figure 11 for the CNN model with BioWordVec embeddings. The yellow line represent the validation accuracy obtained by the model if the most recent batch of Quora questions is added to the training data, which is only preserved if it increases the validation accuracy; the green line represents the validation accuracy of the preserved model after each check: the one obtained with the new questions if the validation increased or the one of the previous step otherwise. It is easy to observe how adding a relatively small percentage of Quora questions (30% and 50% in the shown cases, but smaller in other runs, e.g. 10% or 20%) to the training set leads to increase the validation accuracy, enriching the data and allowing to learn some phrases and usages of vocabulary that can be useful to apply to the domain of the BioASQ challenge. However, trying to incorporate a higher percentage of the Quora questions leads to worse validation results, probably because the original BioASQ questions stop being the majority of the training data, and the increasingly numerous Quora questions are less similar than those in the validation set (and in the test partition), so they are less effective for maximizing the validation accuracy.

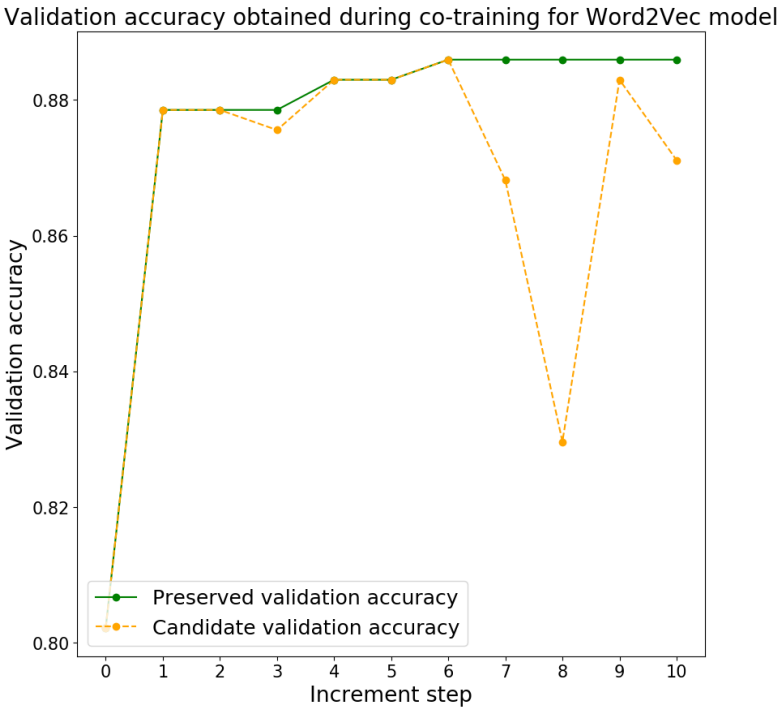

Figure 10: Co-training validation accuracy of the CNN model with Word2Vec embeddings.

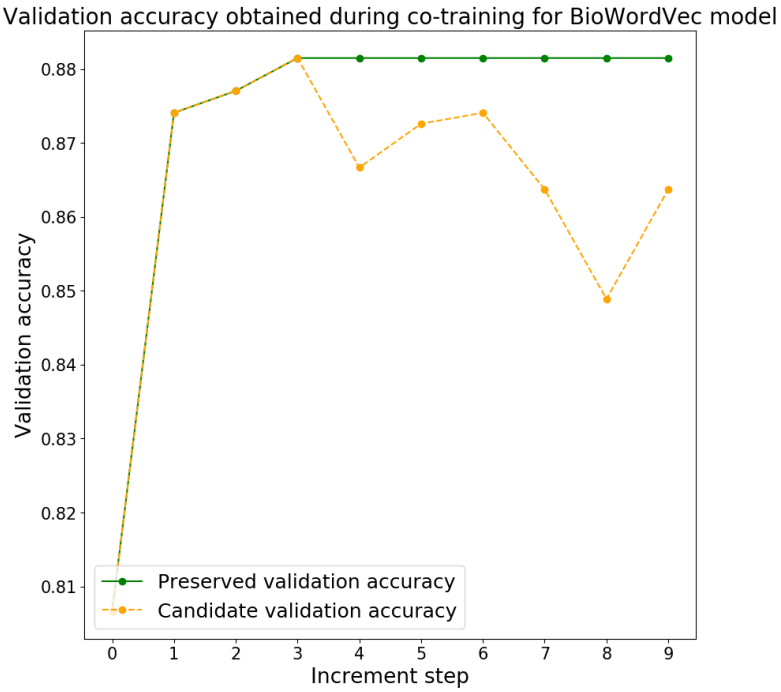

Figure 11: Co-training validation accuracy of the CNN model with BioWordVec embeddings.

As mentioned, the introduction of additional data from Quora did not provide any advantage when using POS-tag embeddings and syntactic-dependency embeddings. In those cases, no improvement is obtained, and a possible explanation lies in the fact that they have a smaller vocabulary, for which embeddings of high quality can be learned even with a small dataset, the BioASQ questions, not finding significant changes when analysing more data, that has very similar patterns of POS tags and syntax even if the vocabulary (in the level of words) is partially different.

Table 8 presents the results of the CNN model with BioWordVec embeddings, which outperformed all the other tested classifiers in both Task 1 and Task 2. This table reflects the usage of the co-training scheme in the model, that allows to increase the F1-score of Task 1 (0.89) up to 0.90. In both tasks, the yes/no classification is perfect, since the model learns a series of (not especially complex) word patterns that distinctively define this question type. The F1-score for the summary class is the same, but the precision and recall are approximately inverted: the precision shows a decrease from 0.87 to 0.83, i.e. there are more false positives in Task 2, probably due to an observed (partial) bias in the classification of Quora questions (by the original model of Task 1) to assign the summary class to some factoid examples, which nevertheless increases the recall from 0.84 to 0.87, since some doubtful instances are classified as summary. The list class has more precision in Task 2 (from 0.86 to 0.93), and it is the only question type for which Quora questions are obviously helpful. A plausible explanation is that the small amount of list instances in the original BioASQ data (as shown in Chapter 2) did not cover some patterns typical in list questions that Quora questions introduce; as examined in Section 3.5.1, there is a high percentage of list questions among the new filtered question pairs, making list the type with the highest proportional increase (in the training data) from Task 1 to Task 2.

Class	**Precision**	**Recall**	**F1-score**
Yes/No	1.0	1.0	1.0
Summary	0.83	0.87	0.85
List	0.93	0.93	0.93
Factoid	0.85	0.81	0.83
Weighted avg	0.90	0.90	0.90

Table 8: Results of the CNN model with BioWordVec embeddings, in Task 2.

5 Conclusions and Future Work

The usage of Convolutional Neural Network models with word embeddings has proved very effective to learn characteristic local word patterns of 4 different question types for the biomedical BioASQ dataset. The most effective combination was the one using word embeddings trained in large corpora of the same (biomedical) domain, BioWordVec embeddings, re-trained with the BioASQ data for greater adaptation, reaching a F1-score of 0.89. Alternative linguistic embeddings, including other types of word embeddings

(Word2Vec and GloVe), as well POS-tag and syntactic-dependency embeddings, were also tried, but were not able to reach the level of semantic suitability of the biomedical word vectors. Other neural architectures were also tried, using different types of recurrent units, but the capabilities of exploiting sequential relations and memory of GRUs and LSTMs was not so effective for the BioASQ data, which is rather small, and the detection of local patterns with convolutions was more appropriate. Unsurprisingly, all neural models outperformed a less complex Machine Learning model (a Support Vector Machine classifier) and a very simple rule-based model, which were not able to obtain remarkable results in the classification task, proving that it was not trivial.

In Task 2, the initially small BioASQ set of questions was expanded with a larger collection of broad-domain Quora questions, that required biomedical filtering to be relevant for the original task. The questions were not manually annotated, but labeled using the previously described classifiers. To minimize the effects of potential error propagation, the questions were divided in batches, added temporarily to the training data to calculate the accuracy on a validation split of the original BioASQ data, and only kept in the training partition if they provided an increase in the validation accuracy, i.e. proved to help to learn interesting patterns to classify the question types. The best model was again the CNN classifier with BioWordVec embeddings, that thanks to the addition of some Quora questions (especially in the originally scarce list type) improved the results slightly, up to a 0.90 F1-score.

Future Work to refine the classifier for the BioASQ challenge may include the addition of more data to learn the different question types, since the best F1-score (0.90) is very good, but far from perfect. The quality and suitability of the new data would be essential, and there would be many options to obtain new instances. Firstly, one should search if there exists any other public dataset of biomedical questions manually annotated with the same criteria, which would be directly usable to expand the training data. Otherwise, if biomedical questions were available but not labeled, one could label them with the current classifier, but the propagation of error may limit the usefulness of the new questions, as it happened with the Quora ones. Alternatively, if there was any dataset with broad-domain questions annotated with the same criteria (and 4 question types) as the BioASQ data, it would be possible to filter them to keep the biomedical ones, i.e. using a binary biomedical text classifier, either an already existing one or a model built from scratch for this task (which, again, would require to obtain annotated data of two types, biomedical or non-biomedical).

References

[1] BioASQ organization team. BioASQ Challenge task 6b, 2018.

[2] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[4] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, 38(3):2758–2765, 2011.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5(Oct):1391–1415, 2004.

[7] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

[8] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[9] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[10] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[11] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):52, 2019.

[12] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*, 2016.

[13] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[14] François Chollet et al. Keras. https://keras.io, 2015.

[15] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.