MoRS At SemEval-2017 Task 3: Easy To Use SVM In Ranking Tasks

Miguel J. Rodrigues[1*] and Francisco Couto[1]

[1]LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
mrodrigues@lasige.di.fc.ul.pt, fcouto@di.fc.ul.pt

Abstract

This paper describes our system, dubbed MoRS (Modular Ranking System), pronounced 'Morse', which participated in Task 3 of SemEval-2017. We used MoRS to perform the Community Question Answering Task 3, which consisted on reordering a set of comments according to their usefulness in answering the question in the thread. This was made for a large collection of questions created by a user community. As for this challenge we wanted to go back to simple, easy-to-use, and somewhat forgotten technologies that we think, in the hands of non-expert people, could be reused in their own data sets. Some of our techniques included the annotation of text, the retrieval of meta-data for each comment, POS tagging and Named Entity Recognition, among others. These gave place to syntactical analysis and semantic measurements. Finally we show and discuss our results and the context of our approach, which is part of a more comprehensive system in development, named MoQA.

1 Credits

This work was supported by FCT through funding of LaSIGE Research Unit, ref. UID/CEC/00408/2013.

2 Introduction

The main difference between Question Answering (QA) and Community Question Answering (cQA) is, while QA systems rely on a user query in order to search and prepare an answer based on the searching capabilities it already has and its documents, in cQA the query and respective related answers are already provided, being only necessary a reordering by relevance of such answers, or perhaps even a rephrasing of such an answer in order to suit better the query.

In recent times, QA systems have attracted great interest in the information retrieval community, and also in the cQA (Höffner et al., 2016).

A characteristic of cQA, is that a user resorts to the web for answers without a given structured knowledge base. The arbitrariness of cQA forums, the dependence and waiting time on their results, may slow the gathering of answers in real time. Also, public forums are dependent of the users input (i.e. answers), which might be rather unstructured, not straight to the point, not related to the question at hand, not well written (i.e. grammatically), lengthy or even incorrect. Our team shares this exact same interest and continuous development in this area.

Our participation focused on the Community Question Answering (CQA) Task 3 SubTask A of 2017 SemEval edition [1] (Nakov et al., 2017), which consisted on reordering a set of comments according to their usefulness in answering the question in the thread. This was made for a large collection of questions created by a user community, provided by the task organizers. Succinctly, Subtask A: Question-Comment Similarity, involved ranking "Good" comments above the "PotentiallyUseful" or "Bad" comments, where there was no distinction between them, since their difference was not important for the task's evaluation method. Finally, the result file was to be a ranked list of the probability of the comments according to their relevance.

We developed MoRS (pronounced "morse"),

[*]Corresponding author: mrodrigues@lasige.di.fc.ul.pt

[1]http://alt.qcri.org/semeval2017/task3/

which went back to simple and rather effective technologies, making it available at a later stage to the public, with minimal pre-requisites and ease of (re)use. MoRS addressed Subtask A of Task 3 by first recognizing relevant terms in each query and also in the respective comments in the thread related to the question being analyzed. Next, the system builds its features by passing through a submodule which analyzes each question and respective comments. Then, MoRS compares named entities from the questions and comments and cross-references them. It also identifies the comments that shared most concepts with the ones associated to the thread question. MoRS employed a semantic similarity to measure how close in meaning both comments and questions are even if they do no share the same exact concepts.(Couto and Pinto, 2013) Additionally, MoRS used Machine Learning (Pedregosa et al., 2011) techniques to classify if a comment as ”Good”, as explained in the description of the Subtask, and ”Not Good” according to the comment's relevance.

The paper also describes the main system where MoRS is part of, which is a larger and modular pipeline, MoQA (Modular Question Answering, pronounced ”mocha”) and also presents the result measure values obtained in Subtask A. Despite successful implementation we did not get the desired results due to data set corruption found only after result submission.

The following section, Section 3, approaches some works that have inspired our system overall, Section 4 explains its composing sub-modules, in Section 5, we present and discuss our results, and finally in sections 6 and 7 we talk about our conclusions and how we plan to approach the future work and applicability of MoRS.

3 Related Work

When coming upon methodology and features to use, the most important ones features came from (Mihaylova et al., 2016), from where we could see such features like: (1) the number of question marks in the documents, (2) whether it contains smileys, e-mails, “thank you” phrases, (3) number of offensive words from a predefined list, (4) length of the answer (in characters), (5) if it includes a first person singular, or (6) plural pronoun, or even (7) if the author of the comment is the same as he author of the question at hand. Moreover, we could use, if available, characteristics such as the position of the comment in the thread, and the ID of the author of the comment. After tokenization, another metric used is the ratio of the comment length and of the question length (in of number of tokens), the number of comments from the same user the thread and the order in which they are written by him. Other aspect of meta-data worthy of exploration is the presence and the number of links in the question and in the comment (inbound or outbound), taking into consideration that the presence of a reference to another resource is indicative of a relevant comment(Mohtarami et al., 2016). These features were the ones we explored in the development of MoRS.

The modularity promoted by OAQA (Yang et al., 2015) and YodaQA (Baudiš, 2015) along with an also modular and reusable and reshapable implementation developed in WS4A (Rodrigues et al., 2016), shaped the idea of a system for lay users, using resources easily available.

4 MoRS Pipeline

As we can see in Figure 1, the system is separated and defined in several modules that work as a pipeline, where each module was designed to be as much independent from the others as possible. The most complex module is the Scorer, which will be detailed in this section. The first step, takes the xml files provided by the organization that go through our xml parser (a). This parser is specific to the format of the xml file, and in principle is the only module that has to be user defined for future use in other projects. The information that comes out of this parsing is the question itself, its author, and from each comment, the author, the text that compose the comment and, if in training phase, the golden score of the comment. The comment then goes through the Scorer module (b), and each comment goes through various scoring methods already described in section 2 which involve, (1) cross-matching of Named Entities (exact and partial tokens), using Stanford Named Entities Recognition (Finkel et al., 2005), determining the number of named entities that the comment has in common with the question; (2) if the author of the comment is the same as the author of the question, giving it away that such a comment would not be fit, since the questioner only on rare occasions answers his own question; (3) if the comment has any question marks, proving that

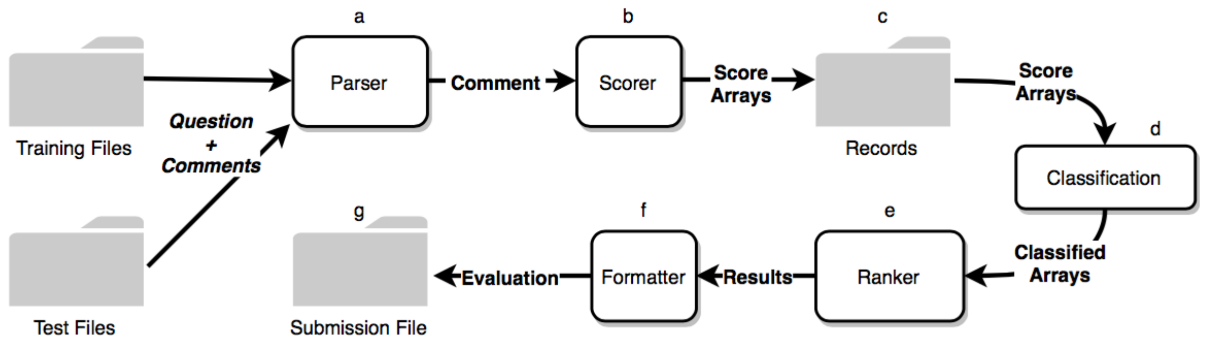

Figure 1: The pipeline of MoRS with its main modules

that comment does not answer the question by not being assertive; (4) if there is any swear words or even (5) misspelled words (from a given list), that may demonstrate a lack of zeal in the answer, plain ignorance or at least lack of effort from the author when answering; (6) sentence semantic similarity score for sentences between the question and the comment, based on the Wu-Palmer metric [2]; (7) if there are any personal pronouns, making the notion of opinion making, which may indicate an answer to the question; (8) the presence of other question URLs or even (9) image URLs which might indicate that a question might be already answered in another thread; (10) the existence of nouns in common may point to similar concepts in discussion in the comment; (11) the presence of smileys, from related work showed that they are not a good sign or assurance in this way, because they did not show seriousness from the comment's author; (12) the number of comments from that author, in which a relative large number indicates that the author has many comments in that thread that at least do not answer correctly the question, therefore the need of other comments. Finally,(13) the length of the comment and it's ratio (14) with the length of the question.

After each set of question plus ten comments, the resulting arrays are written to the Records file (c) for the next step: the classification phase (d): after all questions and correspondent answers are dealt with, we use Machine Learning, more specifically Support Vector Machines (SVM), to classify between "Good" comments and other comments, based on the information provided in the training files.

The "training" phase from the classification ends here, in step (d).

In the second phase, the test files go through the same modules as the training files did, with the difference that in the classification module, the resulting arrays are classified as "Good" or not, and then they go through an implementation of SVM^{light}, SVMrank (e).

SVMrank is an instance of SVMstruct (Joachims, 2002) which mainly features: a fast optimization algorithm, a working set selection based on steepest feasible descent, the ability to handle thousands of support vectors and training examples.

The classified arrays are then transformed in the following format, where the first digit is the importance/relevance of that comment. The larger this first number, more important is the comment, qid denotes the question number, for classification within that question, and the 1:, 2:, 3:, etc. are the scores for each feature. Here follows an example:

```
1 qid:2 1:0 2:0 3:1 4:0.2
2 qid:2 1:1 2:0 3:1 4:0.4
1 qid:2 1:0 2:0 3:1 4:0.1
1 qid:2 1:0 2:0 3:1 4:0.2
```

SVMrank has also a learning phase, where the scored arrays from the training files were provided. After the ranking scores for each question is given, these are run through the Formatter module (f), where the submission file is prepared according to the Subtask's requirements specified in the instructions (g).

5 Results

Our results placed us on the bottom of the table, with the best MAP result belonging to the KeLP team of 88.43, our result of 63.32, and the baseline just slightly lower of 62.30.

Comparing to last year's standards and to our

[2]http://sujitpal.blogspot.pt/2014/12/semantic-similarity-for-short-sentences.html

Submission	MAP	AvgRec	MRR	P	R	F1	Acc
KeLP	79.19	88.82	86.42	76.96	55.30	64.36	75.11
MoRS	***81.15***	***81.42***	***88.44***	***74.37***	***99.94***	***85.28***	***74.34***
Baseline	45.56	65.42	53.50	34.44	76.41	47.47	43.32

Table 1: MoRS' comparison to last year's task 3 results.

Submission	MAP	AvgRec	MRR	P	R	F1	Acc
Beihang	0.714	89.2	77.265	-	-	-	-
MoRS	***79.91***	***80.02***	***86.51***	***74.39***	***100***	***85.31***	***74.39***
Baseline	45.56	65.4	53.50	-	-	-	-

Table 2: MoRS' results for the development set of 2017.

team's surprise, MoRS was far from achieving a comparable performance. After verifying our classification module of "Good" and not "Good" answers, we noticed that our module was defect, lacking about 95% of the arrays necessary to build it, due to a small pipeline error which did not warn us about this mistake, and continued regardless.

After re-running MoRS, and also verifying every step of our system, we compared the results of MoRS with the data-sets from last years' task (Table 1) and with the development set of 2017 (Table 2), and confirmed that in fact, this year's results would have been much better if the models were correctly built.

To our surprise, our scores were significantly lower than what we were experiencing during our scores during the development phase, which achieved very similar results, consisting in a MAP score of 79.91. Highlight to a maximum score of 100 in the Recall.

Another thing we noticed is that the SVMrank had quite the same behavior in both cases, so the issue of the results was exactly the classification of good answers, which brought the results to a surprising first place if it had participated in SemEval 2016 Task3. This was mainly because of our research in the features used by the teams in that year's task and choosing what we thought best fit the purpose of this task. The scores for both 2016 test set and 2017 dev set are available in `https://github.com/migueljrodrigues/MoRS-Scores`.

6 Conclusion

As for this challenge we wanted to go back to simple, easy-to-use, and somewhat forgotten technologies (some going back to 2002) that we think, in the hands of non-experts, could be reused in their own data sets. To simplify the task of future users, we implemented a pipeline, where only the data set provided has some restrictions of format (xml), and we took to a minimum the prerequisites necessary to run the same pipeline (xml files of training and testing), which does not need any computational requisites unavailable to those that do not have large processing resources. The result in this challenge is negative due to a pipeline error. Despite this, after solving the problem, MoRS achieved top results comparing to last year's scores and this year's development set, and so we believe in bringing an easy and simple system to the hands of non-experts in this area, while taking advantage of its capabilities.

7 Future Work

For the algorithm in itself, and since the system is already in place, we pretend to take part in next years Task, maybe with a larger participation in the other subtasks of Task 3. Also, some immediate improvements are to be regarded, such as the handling of empty questions, that represented in the subject, and even use the subject to better grab a context of the question in itself.

In the future, we intend to assimilate this module into MoQA, a Modular Question Answering system, already in development. This system intends to make use of the capabilities developed here in order to rank answers from biomedical articles (from PubMed) taking into account a user query. The system, besides the biomedical example, is to be modular to any domain knowledge, if the correct data-set is to be provided. This system also has the ability to adapt to general questions without a specific domain knowledge, thanks to implementation of DBpedia[3] RDF queries through a Web Service. Finally, MoQA will be easy to understand and reshape according to the users' likeness, if they want to, due to its' modular nature and clear and simple division of the same modules.

References

Petr Baudiš. 2015. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*. pages 1156–1165.

Francisco M Couto and H Sofia Pinto. 2013. The next generation of similarity measures that fully ex-

[3]http://dbpedia.org

plore the semantics in biomedical ontologies. *Journal of bioinformatics and computational biology* 11(05):1371001.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 363–370.

Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. 2016. Survey on challenges of question answering in the semantic web. *Submitted to the Semantic Web Journal* .

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 133–142.

Tsvetomila Mihaylova, Pepa Gencheva, Martin Boyanov, Ivana Yovcheva, Todor Mihaylov, Momchil Hardalov, Yasen Kiprov, Daniel Balchev, Ivan Koychev, Preslav Nakov, et al. 2016. Super team at semeval-2016 task 3: Building a feature-rich system for community question answering. *Proceedings of SemEval* pages 836–843.

Mitra Mohtarami, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Tao Lei, Kfir Bar, D. Scott Cyphers, and Jim Glass. 2016. Sls at semeval-2016 task 3: Neural-based approaches for ranking in community question answering. In *SemEval@NAACL-HLT*.

Preslav Nakov, Doris Hoogeveen, Lluís Márquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Miguel J. Rodrigues, Miguel Falé, Andre Lamurias, and Francisco M. Couto. 2016. WS4A: a biomedical question and answering system based on public web services and ontologies. *CoRR* abs/1609.08492. http://arxiv.org/abs/1609.08492.

Zi Yang, Niloy Gupta, Xiangyu Sun, Di Xu, Chi Zhang, and Eric Nyberg. 2015. Learning to answer biomedical factoid & list questions: Oaqa at bioasq 3b. In *CLEF (Working Notes)*.