Automating the Extraction of Essential Genes from Literature

Ruben Rodrigues[1], Hugo Costa[2], and Miguel Rocha[1(✉)]

[1] CEB - Centre Biological Engineering, University of Minho, Braga, Portugal
mrocha@di.uminho.pt
[2] Silicolife Lda, Braga, Portugal

Abstract. The construction of repositories with curated information about gene essentiality for organisms of interest in Biotechnology is a very relevant task, mainly in the design of cell factories for the enhanced production of added-value products. However, it requires retrieval and extraction of relevant information from literature, leading to high costs regarding manual curation. Text mining tools implementing methods addressing tasks as information retrieval, named entity recognition and event extraction have been developed to automate and reduce the time required to obtain relevant information from literature in many biomedical fields. However, current tools are not designed or optimized for the purpose of identifying mentions to essential genes in scientific texts.

In this work, we propose a pipeline to automatically extract mentions to genes and to classify them accordingly to their essentiality for a specific organism. This pipeline implements a machine learning approach that is trained using a manually curated set of documents related with gene essentiality in yeast. This corpus is provided as a resource for the community, as a benchmark for the development of new methods. Our pipeline was evaluated performing resampling and cross validation over this curated dataset, presenting an accuracy of over 80%, and an f1-score over 75%.

1 Introduction

In recent years, organisms modified genetically have been used as hosts in the production of compounds of interest (e.g. biofuels or drugs) through Biotechnology [1,2]. In many cases, these hosts are subjected to specific genetic modifications to design strains that are able to improve productivity or yield in these bio-processes. The identification of essential genes for these microbial hosts is an important task within this effort.

In this context, several repositories with manually curated information of organism oriented gene essentiality (e.g. OGEE [3], SGD [4]) emerged to identify which genes can be removed maintaining the modified organism viability, given some experimental conditions (e.g. media). The construction of such repositories requires a large amount of information that is spread in different sources, mainly scientific literature [3,4]. The extraction of relevant information from literature

P. Perner (Ed.): ICDM 2018, LNAI 10933, pp. 75–87, 2018.
https://doi.org/10.1007/978-3-319-95786-9_6

about essential genes is a time consuming task, requiring a huge amount of manual curation from researchers.

The Text Mining (TM) field has emerged from the efforts to automate and reduce the required time to retrieve and extract relevant information from literature. This field combines computational approaches applied for several linguistic challenges, as the identification of relevant documents for a specific theme, the recognition of name entities with biological meaning from texts, or the extraction of semantic information of these named entities or relationships/ events relating them [5].

The identification of gene mentions in literature, with a correct association to a specific organism, is a difficult task even when we have a selected set of relevant documents. Named entity recognition (NER) is a TM field that has been used for the identification of biological entities assigned to a class of interest (e.g. proteins, compounds, genes or cell lines) [5–8].

Dictionaries and ontologies have been used as resources for NER, being used as data structures supporting search and matching algorithms (e.g. binary search algorithm), used to annotate free text with specific entries from biological databases and other repositories [6]. This approach has some limitations, being one of them the need to have complete databases for a specific biological context that usually are not enough to match all possible entities (e.g. lack of all possible gene synonyms on a gene dictionary).

NER approaches based on expression rules have been used to match named entity variants, which are not present on dictionaries and ontologies [7]. However, this requires the definition of complex rules created from name patterns for a specific biological context.

Supervised Machine Learning (ML) models, such as Support Vector Machines (SVMs) or Conditional Random Fields (CRFs) have been applied for NER with fast and reliable results [8]. These methods work by training ML models from a large set of data, typically manually curated, which may then be applied to predict the classes of new examples. ML methods have also been used for many other applications in TM.

In the case of NER for genes, we will opt here for the use of dictionaries, since in this particular case they have shown good performance. Indeed, dictionaries for gene names and synonyms for well studied organisms, such as yeast or humans, can achieve good performance in NER tasks. Also, applying a curated dictionary with gene names for a determined organism improves the specificity of the dictionary based NER systems.

When trying to assess gene essentiality from text, a first step of NER to find gene mentions is needed. Then, a second level of information extraction is required to identify if the identified genes are essential or not, given the semantics of the sentence where the gene is mentioned. This will be handled as a problem of Event extraction (EE), another major TM task, which tries to identify semantic interactions between previously annotated entities [3,5,9].

We will consider here that the presence of certain keywords, like “essential”, “nonessential” or “viability”, can be important to define the sentence’s

semantics. We will call these keywords as triggers and identify those in sentences also using NER, matching the tokens in the sentence with a set of pre-defined keywords.

Pairing mentions to triggers and genes can give us information on the semantics of the sentence regarding the gene's essentiality. Here, for each pair, we will provide a classification task seeking to decide if the pair is an essentiality event or not. An event can be defined as a relation between a set of entities with biological interest. In our case, this relationship will be established between a gene and a trigger, and will represent an event of gene essentiality.

As an example, in the sentence *"Disruption of gene A compromises the viability of organism D"*, the entity *"gene A"* and the trigger *"viability"* are associated as a pair, which is classified as an essentiality event *"gene A - viability"* due to the meaning of the whole sentence (the verb *"compromises"* implies that the *"gene A"* is essential for *"organism D"*).

Shallow parsing and dependency parsing methods have been applied to different EE tasks, identifying semantic patterns on sentences that denote the interaction of annotated entities in a sentence [10,11]. Those methods require a set of rules, defined as grammars, that try to represent the structure of a sentence computationally. Those grammars are difficult to obtain because they require large amounts of written text for a specific theme, being normally trained and curated from large text repositories as journal news.

Co-occurrence based methods are able to identify events from associations of entities present in each sentence [12]. Those associations are then classified as event or not regarding a arbitrary rule, a statistical or an ML model. Similarly to ML models applied for NER, ML models applied for EE based on co-occurrences require a large amount of documents manually annotated with events to allow for ML model training.

Training ML models on an EE co-occurrence based system using literature with curated gene essentiality evidences would enable the prediction of novel gene essentiality evidences in unannotated literature. However, previously proposed NER and EE systems lack specificity to extract genes with an essentiality context, since they are designed to identify biological entities and events within a more generic context. Also, there is a lack of adequate manually curated datasets to enable training ML models for this task.

In this work, we aim to help researchers to identify essential gene evidences from literature in an automated way, by proposing a gene essentiality extraction method, which is implemented by a TM pipeline including NER and EE subsystems. Our method firstly identifies gene names for a specific organism and triggers related to gene essentiality, and then provides ML-based models for the classification of the essentiality gene context present in the sentences where gene-trigger pairs are identified. We have also created a large manually curated corpus for yeast genes essentiality with over 6000 sentences, which allows to train our ML models and provide a benchmark for their validation, as well as a resource for the research community.

In this article, we provide a detailed description of the gene essentiality extraction method proposed in this work. Afterwards, the system performance is evaluated in terms of precision, recall and f1 score using our novel curated dataset. Finally, we discuss the advantages and further improvements of the proposed system.

2 Algorithms and Implementation

2.1 Proposed Algorithm

Our system contains two main steps to extract essential genes: (i) identification of triggers and gene names with a dictionary-based NER approach, and (ii) classification of essential genes with an EE based ML approach. Figure 1 shows the overall pipeline with the two main steps of our system.

The NER for gene identification is performed using a dictionary-based approach that requires a dictionary built with a set of gene names for a specific organism (e.g. *S. cerevisiae* or *E. coli*). The system matches and annotates all dictionary names against the free text using a binary search algorithm.

On the other hand, the triggers are recognized following a similar approach, using a dictionary built from a curated set of names like "essentiality", "essential", "nonessential", "non-essential", "unessential gene" or other variation keywords that define the essentiality context of gene names in the sentence.

In the second step of the pipeline, an EE sub-system is used to annotate events between the previously identified genes and triggers. Those events are classified as essential, not essential or not related taking into account the sentence meaning. For EE, we use an algorithm designed for general-purpose tasks, which takes pairs of annotated entities and classifies those accordingly to a set of features.

The EE sub-system is separated in two main pipelines: training and prediction. In both pipelines, there is a common step, the feature matrix generation which requires the MALLET framework [13]. This step processes the corpus and converts it into a matrix of features characterizing pairs of entities (which can be labelled as possible events).

The set of features used in our method is provided and briefly described in Table 1. These features can be divided, regarding the information extracted from texts, into two main groups: sentence morphology and event morphology.

Sentence morphology features are extracted from the sentences' structure, including any information not directly related with the annotated pairs (e.g. sentence length, verbs present in the sentence, etc). On the other hand, event morphology features are related directly with the pair of entity annotations (e.g. words of the possible event, positions of those words in the sentence, etc.).

Entity annotations present in each sentence are paired and treated as possible event instances. For example, consider the sentence *"In contrast to gene A, the gene B is essential for growth."* (Fig. 2), in which *"gene A"*, *"gene B"* and *"essential"* are the annotated entities obtained from the NER module. These

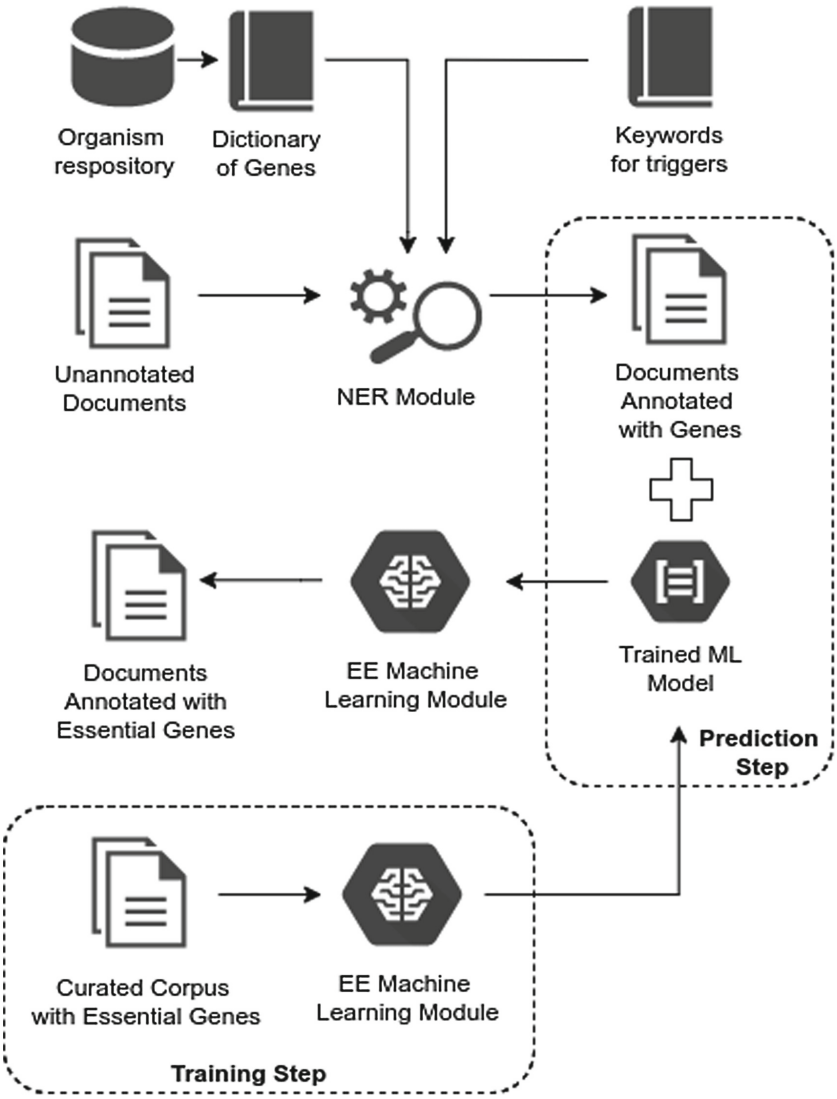

Fig. 1. Overall event extraction pipeline.

annotations are converted into three instances of the matrix considering all possible pairs: *"gene A - gene B"*, *"gene A - essentiality"* and *"gene B - essentiality"* (as observed, the system considers *"gene A - gene B"* as equal to *"gene B - gene A"*).

The EE pipeline contains a filtering approach to select which entity pairs will be used to train ML models. The filtering approach selects which possible pairs of entity classes are allowed in the model (e.g. only pairs that have a trigger and

Table 1. Features used in event classification organized by groups

Feature group	Feature name	Feature description
Sentence morphology	Count tokens between	Counts the number of tokens between two annotations on the event
	Count tokens outside	Counts the number of tokens outside two annotations on the event
	Event annotation between commas	Tests if the annotation event is between commas
	Event annotation between parenthesis	Tests if the annotation event is between parenthesis
	Keyword between event entities	Tests if "not", "for", "by", "in" or "from" keywords exist between event entity tokens
	Lemmas between event annotations	For each lemma, tests if it is present between event token annotations
	Lemmas after keyword until sentence end	For each lemma, tests if it is present in the sentence from keywords "not", "for", "by", "in" or "from" until the end of the sentence.
	Verb between event entities	For each verb, tests if it exists between event annotation tokens
	Verb outside event entities	For each verb, tests if it exists outside event annotation tokens
	Last verb before event	For each verb, tests if it exists before event annotation tokens
Event morphology	Annotation tokens	For each pair of token annotations, checks if it is in the event
	Positions in sentence	For each pair of token annotations, gives the (position/size tokens) in sentence
	Contains token annotations that starts with negative evidence	Tests if annotation starts with "non" or "un" or "in"
	Annotation lemmas	For each pair of lemmas from token annotations, tests if it is in the event
	Annotation part of speech	For each pair of part-of-speech from token annotations, tests if it is in the event
	Event part of speech representative	For each sequence of part-of-speech from all annotations using dependency parsing label nodes, tests if it is in the event
	Event entity annotation type	For each pair of possible entity types, tests if it is in the event
	Dependency tree distance	Minimum distance in number of arcs to get from one entity to other entity in the event

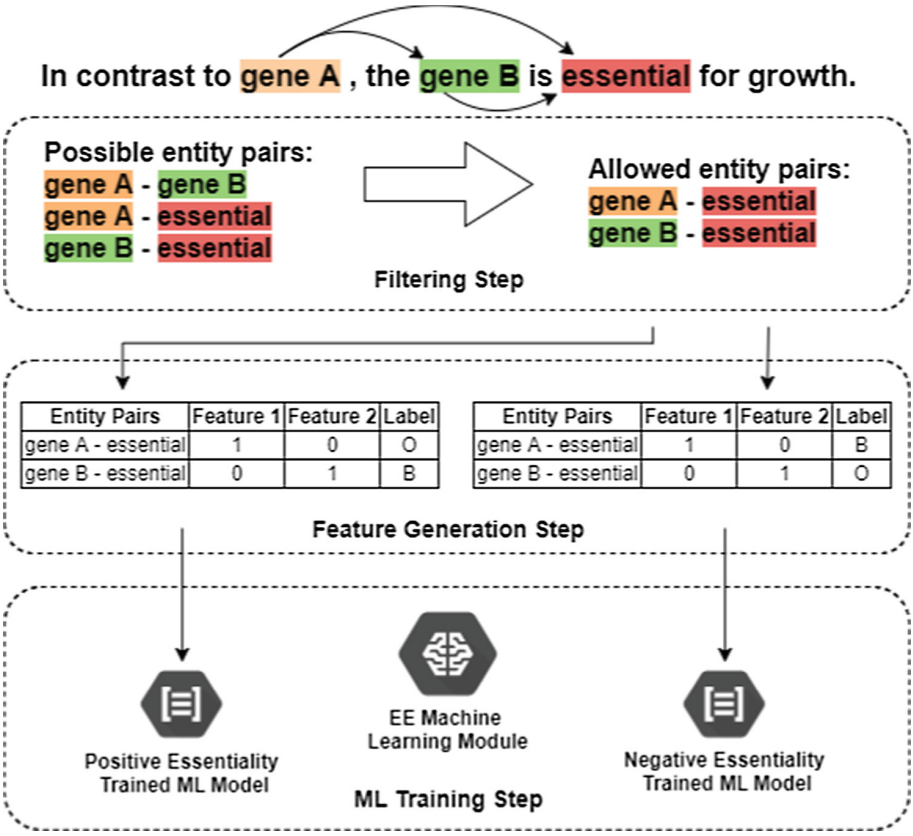

Fig. 2. Sentence example with annotated entities, filtering, feature generation and ML training steps.

gene will be considered in this case). Using the example above, only *"gene A - essentiality"* and *"gene B - essentiality"* are used for the next step.

For each pair selected from the filtering module, a set of features are generated by all modules based on sentence morphology and syntax (Table 1). On the training pipeline, the annotated events are used to label the matrix instances as a binary classification problem:

- "B" for an entity pair that is an event, labelled with the target class of the model;
- "O" for an entity pair that is not an event, or is an event with a type not equal to the target class of the model.

Note that if there are three or more classes in the task, a model for each class is defined. In our case, three models will be trained, for essential, non essential and not related classes.

On the prediction pipeline, each of the ML trained models are applied to a sentence, labeling all pairs of annotations with "B" or "O", as shown on Fig. 2. Each prediction assigns a score to each possible outcome.

When there are three or more classes, the system allows the combination of several trained ML models that are able to predict events with different classes for the same pair. This leads to an overlap from predictions of two or more models which is solved by selecting the event with the highest prediction score.

The ML algorithm in this work uses Support Vector Machines (SVM), as provided by the LibSVM software library [14]. Default LibSVM parameters are used on training (C-SVC with linear kernel, cost equal to 1 and activated estimation of probabilities).

Evaluation of EE ML trained models can be performed by two evaluation methods: cross-validation or resampling. The evaluation system encompasses a pipeline for dataset processing. The processing pipeline requires the MALLET framework, that produces the event matrix described above, in which the cross-validation or the resampling evaluations are applied to the event instances.

For cross-validation evaluations, the matrix is split into k folds (k is defined by the user). Each fold is used to predict event annotations from ML model trained on the remaing folds.

For resampling evaluations, the holdout procedure is applied on the matrix. In each evaluation, a percentage of random instances from the matrix is used as the test dataset. The remaining instances are used to train the ML model.

The event extraction system was integrated into the BioTML framework [15] that is available in @Note2 [16], a general-purpose biomedical TM platform http://anote-project.org/. This system is accessible through a Java application programming interface (API) that is available at https://github.com/biotextmining/machinelearning.

2.2 Manually Curated Corpus for Yeast

An essentiality corpus was created in this work containing 5240 documents related with *S. cerevisiae* randomly selected from PubMed. The corpus was annotated with a dictionary of genes created from the SGD database. Sentences with at least one annotated gene and a trigger like "essentiality", "essential", "nonessential", "non-essential", "unessential gene" or other gene essential variation keywords were extracted from those documents. Each sentence of the corpus was manually annotated with one of three classes: "essential", "not essential" or "not related with essentiality" for each pair of gene-trigger annotations.

As a result, 6339 sentences with 6912 gene and trigger annotations were obtained. Those annotations allowed to create 4412 pairs of trigger-gene annotations, which were then classified and separated in 3 main groups (Table 2).

The corpus is also made available at https://github.com/biotextmining/machinelearning with annotations in the BioNLP 2011 format (A1/A2 format), allowing to fully reproduce the results of this paper, and to benchmark other methods against the same data.

Table 2. Number of event annotations present in the curated corpus

Event group	Number of annotations
Essential gene	1424
Non essential gene	439
Not related essential gene	2549

3 Results

3.1 Event Extraction System Evaluation

The corpus presented in the previous section was used to validate our EE method. We performed the two possible types of evaluation: cross-validation and resampling.

For cross-validation, events from the whole set of documents were randomly split into k folds. In the results presented in this paper, the value of $k = 10$ was used. The cases in each fold were predicted by an ML model trained using the examples in the remaining folds. The predictions were compared against the manually curated event classes using three metrics: precision, recall and f-score, computed for each of the classes.

For resampling tests, 10 runs of holdout were executed. In each run, 2/3 of the examples (events) in the corpus were randomly selected to train the ML model. The remaining events (1/3) were predicted and compared against the manually curated event classes using the metrics mentioned above. This process was repeated 10 times.

The mean scores obtained from evaluations over the full corpus are described in Tables 3 and 4, for the resampling and cross-validation validations. A confusion matrix from one resampling evaluation run on the full corpus is presented in Table 5.

Table 3. Event extraction resampling evaluation mean scores using the full corpus

Essential gene event group	Recall	Precision	F1
Positive	$56.9 \pm 3.3\%$	$68.9 \pm 2.7\%$	$62.3 \pm 2.6\%$
Negative	$36.5 \pm 8.25\%$	$61.2 \pm 6.4\%$	$45.5 \pm 5.9\%$
Not related	$84.2 \pm 2.1\%$	$72.2 \pm 1.7\%$	$77.8 \pm 1.4\%$

As observed, the trained SVM models performed with similar results in both evaluation methods. The values of f1-scores show better results in the classes with more examples (not related and positive), but a low value on the class with less examples (negative). This is probably due to the original unbalance of the dataset.

Table 4. Event extraction cross-validation evaluation mean scores using the full corpus

Essential gene event group	Recall	Precision	F1
Positive	$57.7 \pm 3.5\%$	$68.8 \pm 7.3\%$	$62.6 \pm 7.50\%$
Negative	$40.7 \pm 7.7\%$	$61.3 \pm 18.2\%$	$48.7 \pm 11.4\%$
Not related	$83.8 \pm 3.5\%$	$72.9 \pm 4.4\%$	$78.0 \pm 3.2\%$

Table 5. Confusion matrix of one run of resampling of full corpus

		Predicted event instances by group event class		
		Not Related Essentiality	**Negative Essentiality**	**Positive Essentiality**
Correct event instances by group of event class	**Not Related Essentiality**	728	27	100
	Negative Essentiality	68	59	18
	Positive Essentiality	203	4	255
Model Accuracy		71.2%		

Overall, these models achieved 71% of accuraccy, confirming these suspitions. ML algorithms are highly influenced by the dataset in which they are trained. The results shown that the SVM model was able to predict with high recall and precision on events not related with essentially, because those instances were more frequent than the other two types of events. The ML model predicted negative events with low recall, because the number of examples present in the training corpus was low.

To try to test the results in a balanced dataset, we merged the classes of non-related and negative events. Also, we removed part of the examples in the non related class. As a result, a filtered dataset was created, with 1596 sentences, 3941 entities, 1424 positive events and 1468 not-related/ negative events.

The mean scores obtained in evaluations performed on the filtered corpus are described in Tables 6 and 7. A confusion matrix from one resampling evaluation run on the filtered corpus is described on Table 8.

Table 6. Event extraction resampling evaluation mean scores using the filtered corpus

Essential gene event group	Recall	Precision	F1
Positive	$84.8 \pm 4.3\%$	$76.7 \pm 3.7\%$	$71.7 \pm 2.3\%$
Not related	$68.3 \pm 4.2\%$	$77.1 \pm 6.1\%$	$71.7 \pm 4.5\%$

Table 7. Event extraction cross-validation evaluation mean scores using the filtered corpus

Essential gene event group	Recall	Precision	F1
Positive	$82.4 \pm 14.5\%$	$76.8 \pm 6.9\%$	$79.5 \pm 7.9\%$
Not related	$68.0 \pm 14,2\%$	$76.3 \pm 10.3\%$	$71.6 \pm 8.7\%$

Table 8. Confusion matrix of a run of resampling of the filtered corpus

		Predicted event instances by group event class	
		Not Related Essentiality	**Positive Essentiality**
Correct event instances by group of event class	**Not Related Essentiality**	265	95
	Positive Essentiality	71	406
Model Accuracy		80.2%	

Results show that the corpus balancing resulted in a improvement of overall prediction scores of the ML models, since the algorithm and the parameters used were the same of the previous evaluations.

Positive essentiality events achieved better recall and precision results. This means that our models can, with a high confidence, find sentences that mention essential genes, with precision, recall and accuracy all with values around 80%. This means that ML models can be used to greatly reduce manual curation efforts for this task.

On the other hand, the negative events are harder to discover and to distinguish from non related events. The ability to discover negative events, i.e. sentences were genes are considered non essential, needs trained models with a larger number of examples.

4 Conclusions and Further Work

In this work, we developed an event extraction method, and its implementation in a computational pipeline, that is designed to identify gene annotations and classify them in terms of essentiality for a specific organism. This system is made available in the BioTML plugin, part of @Note2 text mining framework. A manually curated corpus with gene essentiality data was created and made available in this work, allowing to benchmark our method, but also to validate methods proposed by the research community.

The results show that the ML algorithm is highly dependent of the training dataset, because training a ML model with an unbalanced dataset lead to

relatively poor prediction results, mainly in the negative class (events mentioning non essential genes). Balancing the dataset, and considering two classes, an ML model is able to predict with 80% of accuracy and with more than 75% of precision and recall for the positive events.

Deep learning algorithms applied for text mining approaches are emerging with better results than conventional supervised machine learning methods. Further event extraction system improvements could be lead by implementation of deep learning algorithms like long short-term memory networks to automate the feature selection or to identify sentence patterns specific for essential gene identification. Still, the scenario of data scarcity for this task makes this approach more difficult.

Acknowledgments. This work is co-funded by the North Portugal Regional Operational Programme, under the "Portugal 2020", through the European Regional Development Fund (ERDF), within project SISBI- Ref[a] NORTE-01-0247-FEDER-003381.

The Centre of Biological Engineering (CEB), University of Minho, sponsored all computational hardware and software required for this work.

Conflict of Interest. The authors declare they have no conflict of interests regarding this article.

References

1. Guo, D., Zhang, L., Pan, H., Li, X.: Metabolic engineering of Escherichia coli for production of 2-Phenylethylacetate from L-phenylalanine. MicrobiologyOpen **6**(4), e00486 (2017)
2. Yu, T., Zhou, Y.J., Wenning, L., Liu, Q., Krivoruchko, A., Siewers, V., Nielsen, J., David, F.: Metabolic engineering of Saccharomyces cerevisiae for production of very long chain fatty acid-derived chemicals. Nat. Commun. **8**, 15587 (2017)
3. Chen, W.H., Lu, G., Chen, X., Zhao, X.M., Bork, P.: OGEE v2: an update of the online gene essentiality database with special focus on differentially essential genes in human cancer cell lines. Nucleic Acids Res. **45**(D1), D940–D944 (2017)
4. Cherry, J.M., Hong, E.L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E.T., Christie, K.R., Costanzo, M.C., Dwight, S.S., Engel, S.R., Fisk, D.G., Hirschman, J.E., Hitz, B.C., Karra, K., Krieger, C.J., Miyasato, S.R., Nash, R.S., Park, J., Skrzypek, M.S., Simison, M., Weng, S., Wong, E.D.: Saccharomyces genome database: the genomics resource of budding yeast. Nucleic Acids Res. **40**(Database issue), D700-D705 (2012)
5. Shatkay, H., Craven, M.: Mining the Biomedical Literature. Computational Molecular Biology. MIT Press, Cambridge (2012)
6. Gerner, M., Nenadic, G., Bergman, C.M.: LINNAEUS: a species name identification system for biomedical literature. BMC Bioinform. **11**(1), 85 (2010)
7. Gooch, P.: BADREX: In situ expansion and coreference of biomedical abbreviations using dynamic regular expressions. CoRR abs/1206.4, p. 6 (2012)
8. Campos, D., Matos, S., Oliveira, J.: A modular framework for biomedical concept recognition. BMC Bioinform. **14**(1), 281 (2013)
9. Ananiadou, S., Pyysalo, S., Tsujii, J., Kell, D.B.: Event extraction for systems biology by text mining the literature. Trends Biotechnol. **28**(7), 381–390 (2010)

10. Yakushiji, A., Tateisi, Y., Miyao, Y., Tsujii, J.: Event extraction from biomedical papers using a full parser. In: Pacific Symposium on Biocomputing, pp. 408–419 (2001)
11. McClosky, D., Surdeanu, M., Manning, C.D.: Event extraction as dependency parsing. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT 2011, Stroudsburg, PA, USA, pp. 1626–1635. Association for Computational Linguistics (2011)
12. Chun, H., Hwang, Y., Rim, H.-C.: Unsupervised event extraction from biomedical literature using co-occurrence information and basic patterns. In: Su, K.-Y., Tsujii, J., Lee, J.-H., Kwong, O.Y. (eds.) IJCNLP 2004. LNCS (LNAI), vol. 3248, pp. 777–786. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30211-7_83
13. McCallum, A.K.: MALLET: a machine learning for language toolkit (2002). http://mallet.cs.umass.edu
14. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011). http://www.csie.ntu.edu.tw/~cjlin/libsvm
15. Rodrigues, R., Costa, H., Rocha, M.: Development of a machine learning framework for biomedical text mining. In: Saberi Mohamad, M., Rocha, M., Fdez-Riverola, F., Domínguez Mayo, F., De Paz, J. (eds.) PACBB 2016. AISC, vol. 477, pp. 41–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40126-3_5
16. Lourenço, A., Carreira, R., Carneiro, S., Maia, P., Glez-Peña, D., Fdez-Riverola, F., Ferreira, E.C., Rocha, I., Rocha, M.: @Note: A workbench for Biomedical Text Mining. J. Biomed. Inform. **42**(4), 710–720 (2009)