A Query Suggestion Workflow for Life Science IR-Systems

Maria Esch[1,2,*], Jinbo Chen[1], Stephan Weise[1], Keywan Hassani-Pak[2], Uwe Scholz[1] and Matthias Lange[1]

[1]Leibniz Institute of Plant Genetics and Crop Plant Research (IPK), OT Gatersleben, Corrensstr. 3, 06466 Stadt Seeland , Germany, http://www.ipk-gatersleben.de/

[2]Rothamsted Research, Department of Computational and Systems Biology, Harpenden, UK, http://www.rothamsted.ac.uk/

Summary

Information Retrieval (IR) plays a central role in the exploration and interpretation of integrated biological datasets that represent the heterogeneous ecosystem of life sciences. Here, keyword based query systems are popular user interfaces. In turn, to a large extend, the used query phrases determine the quality of the search result and the effort a scientist has to invest for query refinement.

In this context, computer aided query expansion and suggestion is one of the most challenging tasks for life science information systems. Existing query front-ends support aspects like spelling correction, query refinement or query expansion. However, the majority of the front-ends only make limited use of enhanced IR algorithms to implement comprehensive and computer aided query refinement workflows.

In this work, we present the design of a multi-stage query suggestion workflow and its implementation in the life science IR system LAILAPS. The presented workflow includes enhanced tokenisation, word breaking, spelling correction, query expansion and query suggestion ranking. A spelling correction benchmark with 5,401 queries and manually selected use cases for query expansion demonstrate the performance of the implemented workflow and its advantages compared with state-of-the-art systems.

1 Introduction

Due to advances in high-throughput technologies, the amount of data stored in life science databases is growing rapidly. Each day, tens of thousands of new sequence entries appear in EMBL [1] and also the number of sequences in GenBank [2] doubles approximately every 18 months. It is becoming an increasingly difficult and time consuming task for scientists to derive information from those resources and to keep up-to-date even within their own field of research. In this context, information retrieval (IR) is evolving to a key technology to meet the challenges of the big data age.

IR means to extract relevant information from a collection of structured or unstructured documents in heterogeneous databases, and to return the results ordered by relevance [3]. Frequently, gigabytes of text have to be searched, filtered, screened, interpreted and semantically

*To whom correspondence should be addressed. Email: Maria.Esch@ipk-gatersleben.de correlated to the original intention of the query. Web search engines are most common IR applications. There is no need for a user to have knowledge about complex query languages, underlying data structures or data formats. However, for non-expert users of search engines (IR systems) it is not always trivial to define a sensible and correct query which will capture all relevant information. Furthermore, this is complicated by the nature of life science data, which comes along with additional challenges, such as domain specific terminology and a complex vocabulary. Examples are Latin species descriptions, e. g. “Arabidopsis thaliana”, and complex trait descriptions, e. g. “axillary shoot branching”.

Spelling correction and query expansion are two methods which intuitively provide additional information to users while they are typing their query. This can lead to an improved user experience and increase the chance of matching the user’s query to relevant database entries. For most queries, there are hundreds to thousands of documents that contain some or all of the terms in the query. A search engine needs to rank the documents in an appropriate way to give the user the most pertinent ones. The significance of a document with respect to the user’s query is referred to as “document relevance”. Usually, this is unknown and must be estimated from features of the document, the query, the user history or the databases in general [4].

In this paper, a query suggestion workflow will be presented, which was optimized to fit the needs of the life science community. It includes methods for text decomposition, spelling correction and query expansion. We evaluated individual steps of the workflow using custom benchmark and example datasets. To demonstrate applicability and performance of the workflow it was implemented into the life science IR system LAILAPS [5]. A detailed benchmark is discussed based on example queries, which were taken from query logs of life science web information systems and some manually curated use cases. The application of the implementation methods is demonstrated using the query front-end[1] of the life science IR system LAILAPS.

This paper is organized as follows: In section 2, we will provide an overview about state-of-the-art query suggestion systems in life sciences. Section 3 will describe details of our work. Finally, results will be presented and discussed in section 4.

2 Usage of query suggestions in existing Life Science IR systems

Keyword based querying is the most popular way to interact with databases. Therefore, a comprehensive inverse text index is calculated to support fuzzy tokens matching over all indexed database records. This process deprives the potential to interact with the user and to suggest better suited queries. Query suggestions provide a mean to reduce false positive and false negative hits prior to query execution and to increase the usability of the query front-end. An incomprehensive review of query workflows implemented in some popular Life Science IR systems is presented below.

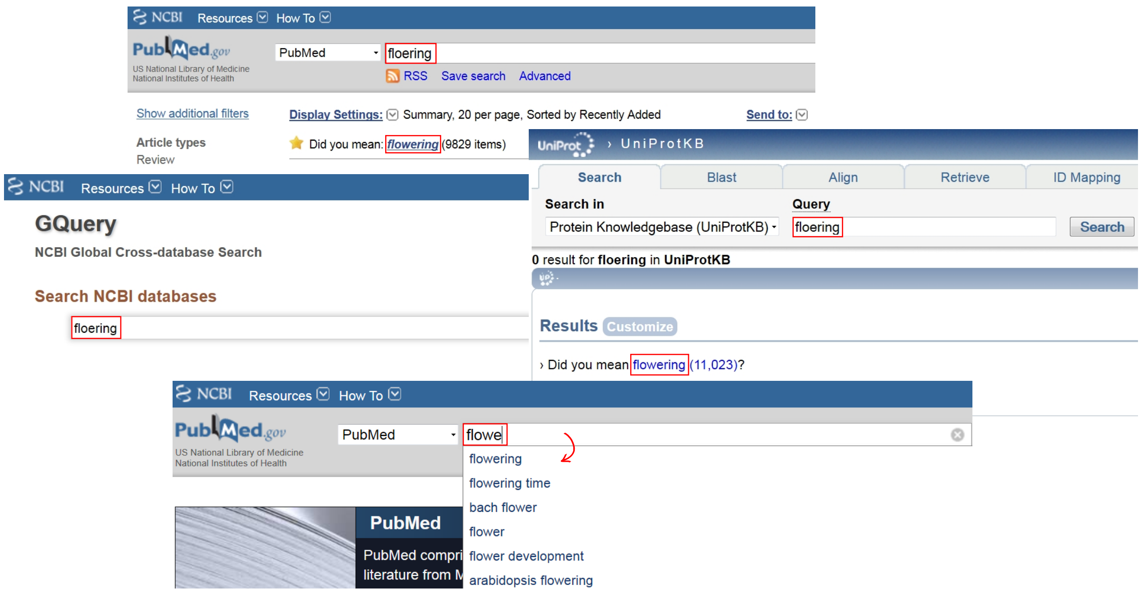
The well-known UniProt database [6] offers a search interface allowing the user to enter a query for finding traits of interest. When the user enters, for example, “flowering time in barley”,

[1] `http://lailaps.ipk-gatersleben.de`

UniProt will return a list of proteins annotated with flowering time. Here, spelling correction is provided, but no query suggestion.

Most search engines in life sciences are designed to search the content of a single database. Therefore, dry lab researchers often have to search in multiple databases until they get their desired information which is time consuming. Entrez GQuery [7] is a search engine, which searches across several NCBI databases[2]. It provides a relevance ranking for the query results, but no query correction or suggestion.

Another well-known search engine for biomedical literature is PubMed [8]. PubMed offers corrections of the user input for smaller misspellings via the “Did you mean” function. However, if there is more than one spelling mistake in a query, mostly no correction can be found. In contrast to UniProt, PubMed provides query suggestion features to facilitate the search (Fig. 1) and ranks the found database results. To find possible suggestions, user history logs are used [9]. This method is often utilised for query suggestions but only works for systems with many users, thus having a sufficient amount of expansion possibilities.



Summary, 20 per page, Sorted by Recently Added

Did you mean: flowering (9829 items)

0 result for floering in UniProtKB

› Did you mean flowering (11,023)?

Figure 1: Screenshots from the 16 of January 2014 ,which show the “Did you mean” function of UniProt, PubMed and Entrez GQuery. The figure also shows the query expansion for PubMed.

Using history logs for query expansions is not an efficient solution for smaller IR systems, but there are also different ways to solve these problems. QTLNetMiner [10] is a web application to search for candidate genes in biological information networks. It provides a query suggester, which proposes alternative queries to users if desired. The query suggester is based on synonym lists derived from domain specific ontologies or dictionaries (e. g. Gene Ontology or Trait Ontology).

The above examples illustrate the usage of query processing and suggestion in life science IR systems and motivate basic recommendations for a more effective query suggestion workflow:

- Support life science specific terminology

[2]The United States National Center for Biotechnology Information advances science and health.

- Spelling correction in combination with word breaking
- Query expansion without user log histories
- Provide query suggestions interactively in real-time

Just as important as the query expansion is the initial spelling correction. If the user is typing a wrong written word, no suggestions in form of expansions will be found. The whole workflow shall be generic and work within any life science IR system regardless of the user or database size.

3 Query Suggestion Workflow

In this work, we present the design and implementation of a query suggestion workflow. It includes tokenisation, word breaking, spelling correction, expansion and ranking. The proposed workflow is illustrated in Fig. 2 and a detailed description of each step in the suggestion process is presented below.

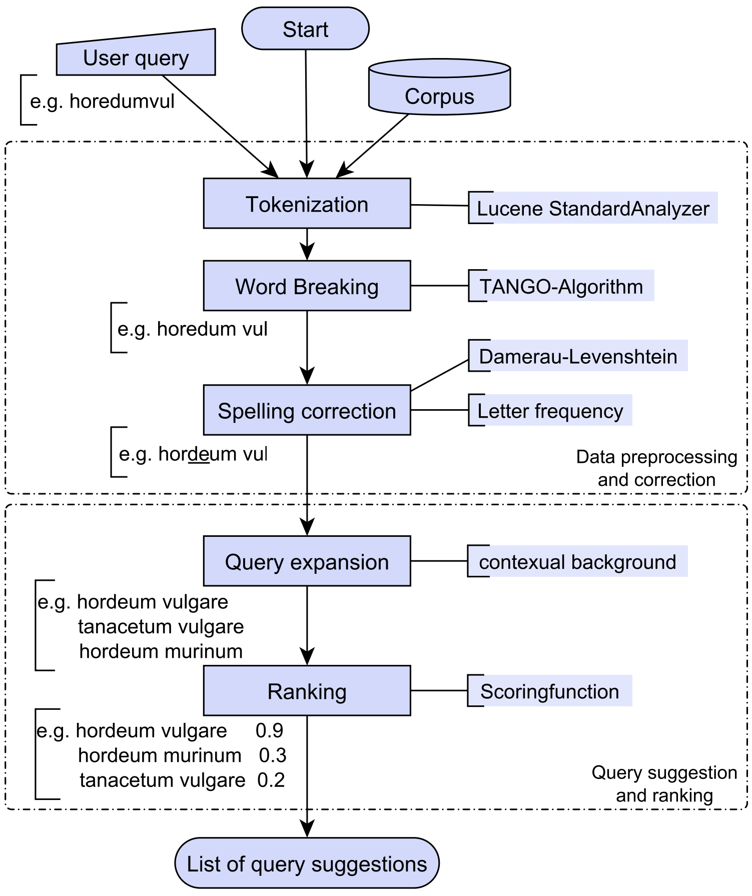


Figure 2: Workflow of a query expansion divided into the two steps of data processing and query suggestion. The example is based on possible hits for the request *horedumvul*.

3.1 Query Segmentation

Query segmentation is the process a user query is split into its individual components also called tokens. Tokens are significant words or other units in a document, which are the basis for text comparisons. In many written languages whitespace character are used as delimiters between two words. However, for life science data some special cases have to be considered, e. g. reaction formulas, citations or identifiers. Apache Lucene[3] a common text indexing system, splits words based on rules from the Unicode Text Segmentation algorithm and applies special enhancements to whitespace-based tokenization. It also include special filter like the *LowerCaseFilter* which is used to create a uniform case-sensitive style to find all matches between the query tokens and those from the corpus.

Lucene's standard tokenizer shows some weaknesses for life science data. For example, the query *Dnapolymerase-Chainreaction:PCR* would be split into the tokens *dnapolymerase*, *chainreaction* and *pcr*. Here, it is obviously helpful to apply a word breaking algorithm after the word segmentation, which will also separate the words *dna* and *polymerase*.

3.2 Word Breaking

A word breaking algorithm can be used to segment composed words or other phrase units which are written together by mistake. Many simple algorithms are based on dictionaries. A widely used method is described by Norvig [11]. In this method, a probabilistic model is built by looking up created n-grams from the user query in a dictionary of words and their corresponding frequencies. This method requires a very large dictionary to cover as many cases as possible. Therefore, we decided for a mostly-unsupervised statistical algorithm. Some languages like Japanese have no delimiters, such as a whitespace, between words. Therefore a word breaking algorithm is needed to segment words in a text. Ando et al. published a method named TANGO (Threshold And maximum for N-Grams that Overlap) to solve this problem without a simple dictionary look up [12]. TANGO checks the boundary of two adjacent n-grams in comparison with their overlapping n-grams. This allows separating compound words like *dnapolymerase* into *dna* and *polymerase* (Fig. 3).

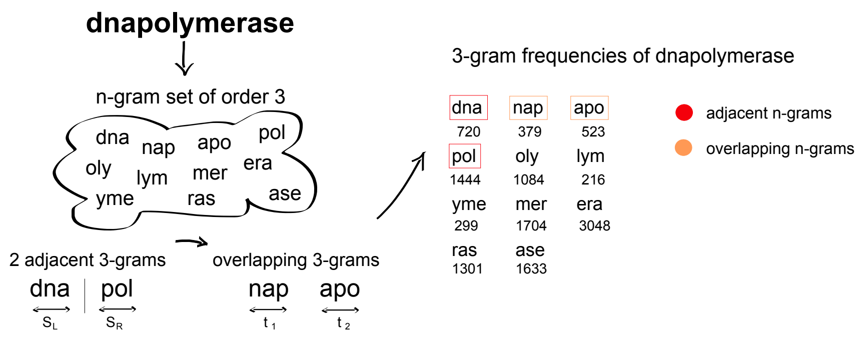


Figure 3: Illustration of the TANGO algorithm for word breaking.

[3]http://lucene.apache.org/

TANGO requires a training set to learn at which position a word has to be separated. The training set can be derived from application case specific dictionaries. We decided to use WordNet[4] which is a lexical database and ontology data sets from the OBO foundry[5] to train the algorithm. This will train TANGO to give stronger consideration to biological words or chemical compound names.

3.3 Spelling Correction

After the phrase pre-processing, possible spelling errors have to be found and corrected before the query can be expanded. Beside phonetic algorithms like Soundex [13] which is based on the sound of the English language, there are also many distance measures for spelling correction. The Damerau-Levenshtein distance [14, 15] measures the similarity between two words and is language independent in contrast to phonetic algorithms. This is important for searching terms in life sciences where often mixtures of different languages like English and also Latin technical terms are used. Soundex works well for English texts, but will probably fail for plant names or chemical reactions. The use of the Damerau-Levenshtein distance seems to be the better choice here. To weight the edit operations, heuristic observations of misspellings [16] and letter frequencies will be additionally included. In this way, beside to phonetic errors and typing errors also different aspects of linguistics were taken into account.

Let $p, q \in S$ denote two distinct nonempty strings $S = (a_{l_1}, \ldots, a_{l_i}, a_{s_1}, \ldots, a_{s_j})$ where $a_l \in A_L$ is a set of letters with $A_L = \{\text{'a'}, \ldots, \text{'z'}\}$ and $a_s \in A_S$ is a set of non-letters. The edit distance of p and q is computed using the enhanced cost function $f_{cost}(p_i, q_j)$. This function uses four scoring matrices: S_D (deletion), S_I (insertion), S_S (substitution) and S_T (transposition). The confusion matrices are taken from [16] and normalised over the number in one matrix. Moreover, $freq(a)$ is the relative frequency of the letter a in the English language [17].

$$f_{cost}(p_i, q_j) = 1 - \begin{cases} S_D(p_{i-1}, q_i) * (freq(p_{i-1}) + freq(q_i)) & \text{deletion} \wedge i > 1 \\ S_I(p_{j-1}, q_i) * (freq(p_{j-1}) + freq(q_i)) & \text{insertion} \wedge j > 1 \\ S_S(p_i, q_j) * (freq(p_i) + freq(q_j)) & \text{substitution} \\ S_T(p_{i-1}, p_i) * (freq(p_{i-1}) + freq(p_i)) & \text{transposition} \wedge i > 1 \\ 0 & p_i \vee q_j \in A_s \wedge else \end{cases} \quad (1)$$

We score non-letter characters with 1 since there is no known sufficient statistic for special characters.

3.4 Query Expansion

Query expansion is a challenging task and different methods have previously been developed using query logs, dictionaries or statistical methods to expand an incomplete query [18]. The

[4]`http://http://wordnet.princeton.edu/`
[5]`http://www.obofoundry.org/`

intention of this work was to integrate a log-free expansion method to make the query suggestion workflow applicable to IR systems with none or limited user logs. Additionally we were interested in a query suggestion that not only completes the incomplete word but also expands the query with extra word groups derived from the text corpus. A method to suggest queries without using log histories was published in Bhatia et al. [19] and has an advantage over existing methods as it not only considers single tokens but also surrounding words from the text corpus. The user query will be divided into two parts, a query background and a last (sub) word. Then, two steps will be performed. First, possible completions to the last sub word will be searched for. Second, the contextual background will be considered to find possible expansions. We modified the algorithm to hide expansions that contain words occurring already in the users query. Otherwise, the user could also get suggestions like *barley hordeum vulgare barley* for the query *barley hordeum vul*.

3.5 Ranking

At the end, all possible query suggestions have to be ranked. The assumption is that the probability $P(p_s \rightarrow R_i^{rel})$ for a suggested query to result in a relevant database record $R_i^{rel} \subseteq D$ is influenced by the similarity of query phrases $p_i \in P$ and suggested queries $p_s \in P$, as well as the frequency of the suggested phrase in the indexed database records.

To compute the query similarity $S(p_i, p_s)$ to a suggested phrase, we calculate the Damerau-Levenshtein distance DL and the Dice [20] coefficient DC. Here, $DL(p_i, p_s)$ is the minimum number of operations ϵ needed to transform one string into the other, where an operation is defined as an insertion ϵ_i, deletion ϵ_d, or substitution ϵ_s of a single character, or a transposition ϵ_t of two adjacent characters. The cost for transformation operations $cost(\epsilon)$ are defined as described in section 3.3. To enable a reasonable computational complexity, we restrict $DL \leq 3$. Next, the Dice coefficient DC and finally the similarity $S(p_i, p_s)$ is calculated, where $S(p_i, p_s)$ is the difference between $DC(p_i, p_s)$ and $DL(p_i, p_s)$. The next factor which has to be computed is the relative frequency of the suggested phrase in the indexed database records $TF(p_s, D)$. Since the final relevance ranking of a database record is computed by the LAILAPS ranking algorithm, only a binary frequency $f(p_s, d_x)$ is used.

$$f(p_s, d_x) = \begin{cases} 1 & p_s \text{ has an exact match in } d_x \in D \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In order to normalise this number, the relative frequency over all indexed database records is computed. Furthermore, we count multiple hits per database record as one hit.

$$TF(p_s, D) = \frac{\sum_{x=1}^{|D|} f(p_s, d_x)}{|D|} \quad \Bigg| \, d_x \in D \tag{3}$$

The final probability can be computed as follows.

$$P(p_i \rightarrow R_i^{rel}) = \lambda * S * TF \tag{4}$$

The factor λ is used to calibrate the weight of text corpus and history logs in D. In order to support a collaborative query suggestion, we set a higher evidence to phrases from the query history than from the database corpus. A manual valuation of the quality of the query suggestions could be performed in the future by tuning λ.

4 Results and Discussion

In the following, we present our results of the implemented workflow and evaluate the obtained results. The implementation of the entire workflow into LAILAPS is subject to current work, but several components have already been included, such as the spelling correction. In order to test the performance of the workflow, the whole query refinement process was successfully set up in a own test environment where each step is implemented as a own method.

Apache Lucene's standard analyzer which is based on the standard tokenizer enabled us to pre-process and to split text into tokens based on different roles. Table 1 shows some typical example queries and how they were pre-processed by Lucene.

Original	Token
1-amino-cyclopropane-1-carboxylate	1-amino cyclopropane-1-carboxylate
http://www.uniprot.org/terms	http www.uniprot.org terms
NCBI_TaxID=332058	ncbi taxid 332058
ID 025R_IIV3	id 025r_iiv3
DOI=10.1006/viro.2001.0963	doi 10.1006/viro.2001.0963
(6E)-8-hydroxygeraniol + 2 NADP(+) = (6E)-8-oxogeranial + 2 NADPH.	6e 8-hydroxygeraniol 2 nadp 6e 8-oxogeranial 2 nadph
H -> Y (in Ref. 2; CAD43308)	h y ref 2 cad43308
fig—222523.1.peg.954	fig 222523.1.peg.954

Table 1: Working method of the Apache Lucene *StandardAnalyzer* shown by typical examples from the area of life science. Left shows the orignal phrases and right showes the decomposed phrases by Apache Lucene.

Lucene uses specific rules for text segmentation which in general provide good delimiters also in life science. However, an issue that is not easy to resolve is the handling of chemical formulas and reactions. The definition of tokens having a chemical background is challenging. Usually, many signs which have a special meaning in reactions, are delimiters in general tokenisers. For example, the shown reaction in Table 1 is separated into eight tokens without a relation between them. This shows that additional rules need to be considered in life sciences.

The word segmentation was done using the TANGO algorithm which allows to correct typos, most likely in form of terms inadvertently written in one word, and also to decompose compound words into their simplexes. A test set was created by using the *GO Biological Process* and removing all whitespaces between GO terms, to test if the TANGO algorithm can accurately break biological terms at the correct positions. The accuracy of the word breaking was tested using different distance measures such as the Dice coefficient indicating the similarity of two terms. An example of the pre-processed original GO terms and the TANGO output are shown in Fig. 4. TANGO was tested with different word lists, n-gram orders and thresholds and the best result that we could observe had a Dice coefficient of 0.404, where 1 means a perfect restoration of the original GO terms. We compared TANGO with other existing methods, such as the Microsoft Web N-gram tool [21] which achieved a Dice coefficient of 0.756 for the same dataset.

autophagic vacuole assembly	autophagic vacuole assembly
autophagic vacuole fusion	autophagic vacuole fusion
rieskeironsulfur protein	rieske iron sulfur protein
peptid yl transferase activity	peptidyltransferase activity
trnabinding	trna binding
ureacycle	urea cycle
ureacycle inter mediate metabolic process	urea cycle intermediate metabolic process
citrulline metabolic process	citrulline metabolic process
argininosuccinate metabolic process	argininosuccinate metabolic process
ribosomal subunitexport from nucleus	ribosomal subunit export from nucleus

Figure 4: An excerpt of the test data which shows the word breaking results compared with the original file. Correctly segmented terms are labelled blue and wrong segmentations are labelled red Orange words are not completely identical with the original text but also not wrong.

The spelling correction algorithm of Section 3.3 was benchmarked using 5,401 pairs of spelling corrected words [22]. To create this benchmark set we analysed a one-year query log of the TAIR database [23] a comprehensive information system for the genome of the model plant *Arabidopsis thaliana*. We assumed that the last query in a session is the correct one from a user's perspective. In order to filter those queries, which were likely refined by a user due to spelling problems, those words are paired which were logged in the same user session, have an edit distance of 1 or 2 and the word length of both is at least 4. Our spelling correction algorithm was able to predict the correct pair in 2,741 (50.7%) cases. Figure 5 shows the LAILAPS query front-end and a summary plot of the results of the automatic spelling correction.

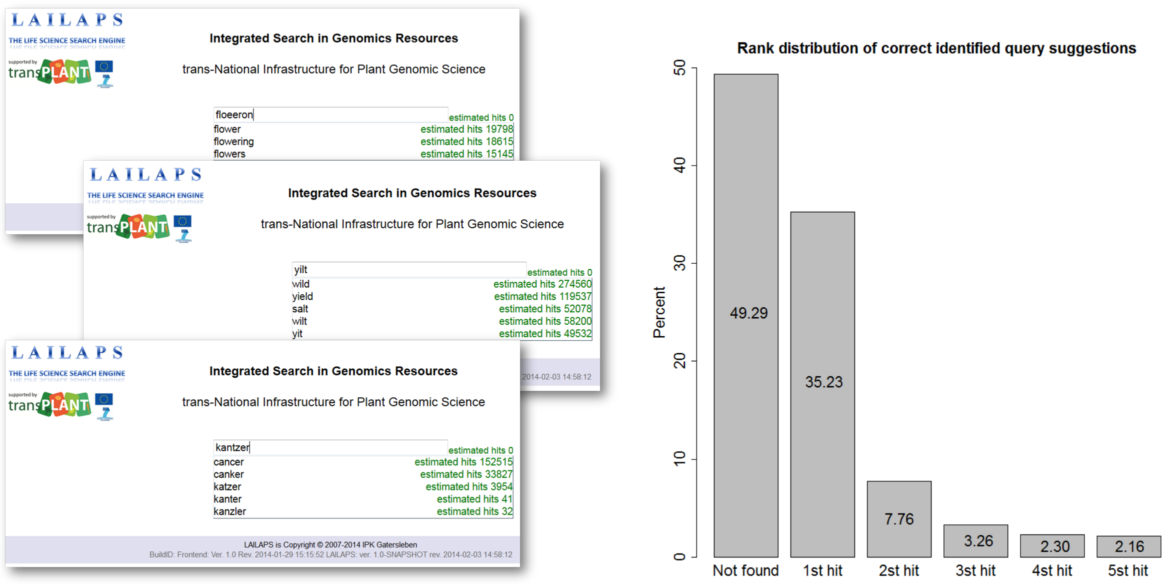
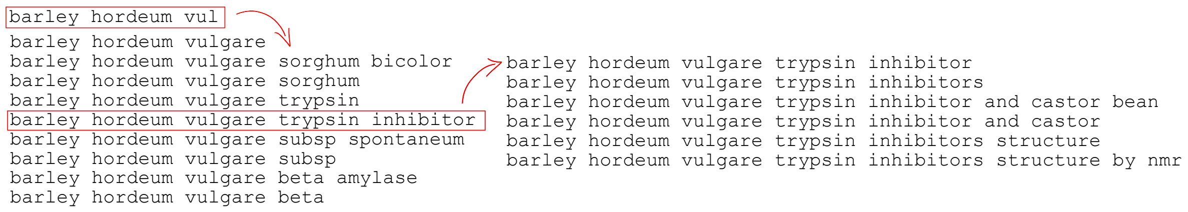


Figure 5: The positive effect of the enhanced spelling correction in LAILAPS is shown by three misspelled keywords: (a) floeeron = flowering represents a typical misspelling caused by hitting neighbouring keys at the keyboard, (b) yilt = yield shows the effect of spelling errors and (c) kantzer = cancer demonstrates language based spelling problems. – The histogram shows the distribution of 5,401 corrected words binned by their position in the suggestion list. For half of all words correct suggestions where under the top five and a fraction of 35% accurately spell corrected words is at the top suggestion.

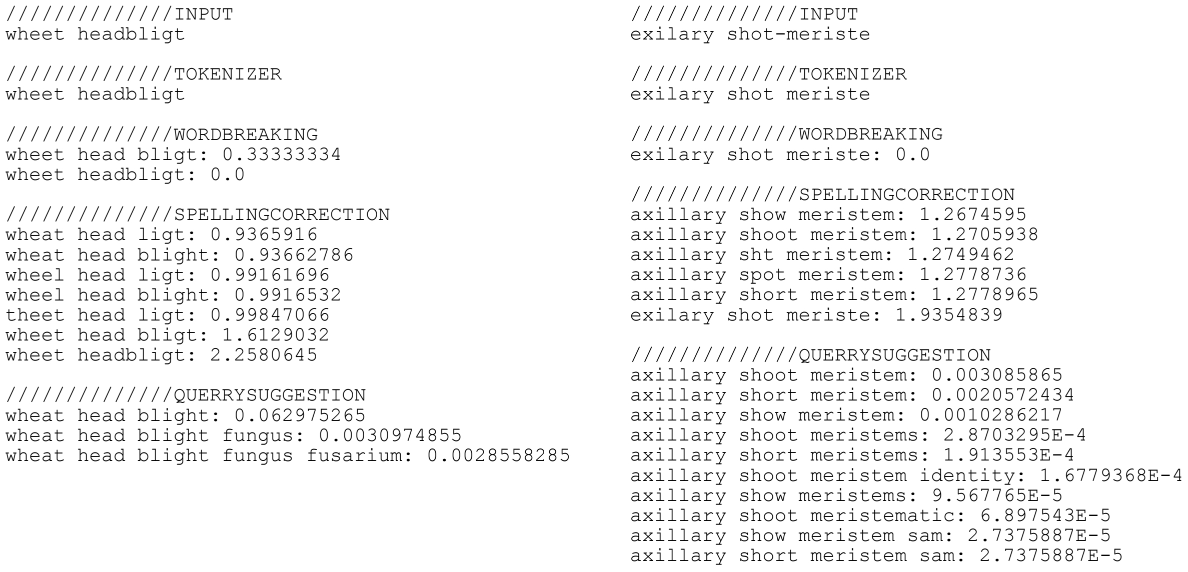
The next step in the query suggestion workflow is the query expansion. Its performance has been assessed using a small test corpus of 35,000 Uniprot entries. We have chosen some typical query examples and evaluated the predicted query expansions. For example, given an incomplete query such as *barley hordeum vul*, our method produces the phrase *barley hordeum vulgare trypsin inhibitor* as one of the top suggested results because it has a high frequency in our corpus. This suggested query can be processed a second time. The results of both query suggestions are shown in Fig. 6. The suggestion *barley hordeum vulgare leaf specific dna binding* led to the Uniprot entry *2SS_RICCO*. It includes a reference to the article *"Structural relationship between barley (Hordeum vulgare) trypsin inhibitor and castor-bean (Ricinus communis) storage protein"* and the following word group: *"BLOCKAGE OF N-TERMINUS, AND NOMENCLATURE. SIMILARITY TO PROTEINASE INHIBITORS. STRUCTURE BY NMR OF 36-156, AND DISULFIDE BONDS"*.


```
barley hordeum vul
barley hordeum vulgare
barley hordeum vulgare sorghum bicolor
barley hordeum vulgare sorghum
barley hordeum vulgare trypsin
barley hordeum vulgare trypsin inhibitor
barley hordeum vulgare subsp spontaneum
barley hordeum vulgare subsp
barley hordeum vulgare beta amylase
barley hordeum vulgare beta

barley hordeum vulgare trypsin inhibitor
barley hordeum vulgare trypsin inhibitors
barley hordeum vulgare trypsin inhibitor and castor bean
barley hordeum vulgare trypsin inhibitor and castor
barley hordeum vulgare trypsin inhibitors structure
barley hordeum vulgare trypsin inhibitors structure by nmr
```

Figure 6: An example which shows the possibilities of a query suggestion using the text corpus as textual background. The first phrase is *barley hordeum vul* and the second is *barley hordeum vulgare trypsin inhibitor*.

To present the potential of the whole workflow, two possible queries were chosen to show the results of each step. Fig. 7 demonstrates the effectiveness of a suggestion workflow and the importance of each step to get a good query expansion.


```
//////////////INPUT
wheet headbligt

//////////////TOKENIZER
wheet headbligt

//////////////WORDBREAKING
wheet head bligt: 0.33333334
wheet headbligt: 0.0

//////////////SPELLINGCORRECTION
wheat head ligt: 0.9365916
wheat head blight: 0.93662786
wheel head ligt: 0.99161696
wheel head blight: 0.9916532
theet head ligt: 0.99847066
wheet head bligt: 1.6129032
wheet headbligt: 2.2580645

//////////////QUERRYSUGGESTION
wheat head blight: 0.062975265
wheat head blight fungus: 0.0030974855
wheat head blight fungus fusarium: 0.0028558285
```

```
//////////////INPUT
exilary shot-meriste

//////////////TOKENIZER
exilary shot meriste

//////////////WORDBREAKING
exilary shot meriste: 0.0

//////////////SPELLINGCORRECTION
axillary show meristem: 1.2674595
axillary shoot meristem: 1.2705938
axillary sht meristem: 1.2749462
axillary spot meristem: 1.2778736
axillary short meristem: 1.2778965
exilary shot meriste: 1.9354839

//////////////QUERRYSUGGESTION
axillary shoot meristem: 0.003085865
axillary short meristem: 0.0020572434
axillary show meristem: 0.0010286217
axillary shoot meristems: 2.8703295E-4
axillary short meristems: 1.913553E-4
axillary shoot meristem identity: 1.6779368E-4
axillary show meristems: 9.567765E-5
axillary shoot meristematic: 6.897543E-5
axillary show meristem sam: 2.7375887E-5
axillary short meristem sam: 2.7375887E-5
```

Figure 7: Results of each step in the suggestion workflow for the incorrect and not clearly formulated phrases *wheet headbligt* and *exilary shot-meriste*. Despite useful suggestions could be found.

These examples underline the effectiveness of query suggestions and demonstrate the value for the user in terms of getting relevant results in a shorter time. Improvements in the area of word breaking are possible and necessary. The TANGO algorithm was able to restore 40% of the original benchmark set compared to 76% using the Microsoft Web N-Gram tool. Expanding the Damerau-Levenshtein distance to heuristic observations of misspellings and letter frequencies allows a detailed view on different error types. The proposed query expansion provides contextual suggestions and is especially suited for IR-Systems without user logs. In addition, the use of life science specific ontologies as synonym lists could provide further improvements for query expansions. There are many aspects in the field of query suggestions which have to be considered. This workflow shows the most important aspects and possible solutions of query suggestion for IR systems in life science which can be easily extended by further techniques.

Acknowledgements

This work was supported by the European Commission within its 7th Framework Programme (thematic area “Infrastructures”), contract number 283496, and the Leibniz Institute of Plant Genetics and Crop Plant Research (IPK).
ME and KHP were supported through the strategic funding given to Rothamsted Research by the Biotechnology and Biological Sciences Research Council (BBSRC).

References

[1] EMBL statistics. `http://www.ebi.ac.uk/ena/about/statistics`, 2014. As of 2014-02-16.

[2] GenBank statistics. `http://www.ncbi.nlm.nih.gov/genbank/statistics`, 2014. As of 2014-02-16.

[3] M. Lange, R. Henkel, W. Müller, D. Waltemath and S. Weise. Information Retrieval in Life Sciences: A Programmatic Survey. In M. Chen and R. Hofestädt (editors), *Approaches in Integrative Bioinformatics – Towards the Virtual Cell*, pages 73–109. Springer Berlin Heidelberg, 2014.

[4] H. Zaragoza and M. Najork. Web search relevance ranking. In L. Liu and M. T. Özsu (editors), *Encyclopedia of Database Systems*, pages 3497–3501. Springer US, 2009.

[5] M. Lange, K. Spies, J. Bargsten et al. The LAILAPS Search Engine: Relevance Ranking in Life Science Databases. *Journal of Integrative Bioinformatics*, 7(2):e110, 2010.

[6] A. Bairoch, R. Apweiler, C. H. Wu et al. The Universal Protein Resource (UniProt). *Nucleic Acids Research*, 33(suppl_1):D154–D159, 2005.

[7] G. D. Schuler, J. A. Epstein, H. Ohkawa and J. A. Kans. Entrez: Molecular biology database and retrieval system. In R. F. Doolittle (editor), *Computer Methods for Macromolecular Sequence Analysis*, volume 266 of *Methods in Enzymology*, pages 141–162. Academic Press, 1996.

[8] J. McEntyre and D. Lipman. PubMed: bridging the information gap. *Canadian Medical Association Journal*, 164(9):1317–1319, 2001.

[9] Z. Lu, W. J. Wilbur, J. R. McEntyre, A. Iskhakov and L. Szilagyi. Finding Query Suggestions for PubMed. In *AMIA Annual Symposium Proceedings*, pages 396–400. 2009.

[10] K. Hassani-Pak. QTLNetMiner. `http://ondex.rothamsted.ac.uk/QTLNetMiner`. A tool for the discovery of candidate genes controlling complex traits.

[11] P. Norvig, T. Segaran and J. Hammerbacher. Natural language corpus data. In *Beautiful Data*, pages 219–242. O'Reilly media, 2009.

[12] R. K. Ando and L. Lee. Mostly-unsupervised statistical segmentation of japanese kanji sequences. *Natural Language Engineering*, 9:127–149, 2003.

[13] R. Russel. Index. US patent 1.261.167, 1918. `http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=1261167`.

[14] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, 1964.

[15] V. Levenshtein. Binary Codes of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[16] M. D. Kernighan, K. W. Church and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics - Volume 2*, COLING '90, pages 205–210. Association for Computational Linguistics, 1990.

[17] R. E. Lewand. *Cryptological Mathematics (Classroom Resource Materials)*. The Mathematical Association of America, 2000.

[18] H. Cui, J.-R. Wen, J.-Y. Nie and W.-Y. Ma. Query expansion by mining user logs. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):829–839, 2003.

[19] S. Bhatia, D. Majumdar and P. Mitra. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 795–804. ACM, 2011.

[20] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[21] K. Wang, C. Thrasher and B.-J. P. Hsu. Web scale nlp: a case study on url word breaking. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 357–366. ACM, New York, NY, USA, 2011.

[22] J. Chen, M. Esch and M. Lange. Spelling correction of TAIR queries using query user refinement which has been recorded in query logs, 2014. URL `http://dx.doi.org/10.5447/IPK/2014/1`.

[23] P. Lamesch, K. Dreher, D. Swarbreck, R. Sasidharan, L. Reiser and E. Huala. *Using The Arabidopsis Information Resource (TAIR) to Find Information About Arabidopsis Genes*, chapter 1, pages 1.11.1–1.11.51. Current Protocols in Bioinformatics. John Wiley & Sons, Inc., 2002.