Matching documents and summaries using key-concepts

Sara Tonelli, Emanuele Pianta
Fondazione Bruno Kessler, Via Sommarive 18, Povo (Trento) Italy
satonelli@fbk.eu, pianta@fbk.eu

Résumé. Nous présentons une méthodologie pour trouver le meilleur appariement d'articles scientifiques et de résumés en trois étapes : la première étape consiste à extraire d'un document une série de concepts-clé, pour donner une représentation concise de son contenu. Ensuite, la liste de concepts-clé est associée à chaque résumé en assignant un score de similarité inspiré par les critères de mesure standard : Précision, Rappel et F1. Enfin, étant donné un graphe biparti pondéré qui représente tous les possibles paires document-résumé, on exécute un algorithme qui trouve le meilleur appariement. Nous évaluons notre approche sur le corpus de test de DEFT, et montrons qu'il obtient de bons résultats (la configuration la plus performante obtient 0.99 F1). Nous exposons en détail plusieurs avantages de cette approche. Par exemple, il ne requiert pas de corpus d'entrainement, mais uniquement un corpus de développement pour le réglage des paramètres. En plus, l'utilisateur peut configurer facilement le type de concepts-clé à extraire, selon des paramètre allant de leur longueur maximale à leur degré de spécificité.

Abstract. We present a methodology to find the best document - summary match based on three steps : first, a set of key-concepts is extracted from the document in order to give a concise representation of its content. Then, the key-concept list is compared with each abstract by assigning a similarity score inspired by the standard metrics of Precision, Recall and F1. Finally, an algorithm is run that, given a weighted bipartite graph representing all possible document - summary pairs, finds the best matches. We evaluate our approach on the DEFT test data and we show that it achieves good results, with 0.99 F1 in the best performing configuration. We also detail some advantages of this approach. For example, it does not require training data except for a development set for parameter tuning. Besides, the user can easily configure the type of key-concepts to be extracted, from their maximum length to the degree of specificity.

Mots-clés : Extraction de concepts-clé, mesures de similarité, appariement de graphes.

Keywords: Key-concept extraction, similarity metrics, graph-matching techniques.

1 Introduction

Key-concepts are simple words or phrases that provide an approximate but useful characterization of the content of a document and offer a good basis for applying content-based similarity functions. In general, key-concepts can be used in a number of interesting ways both for human and automatic processing. For example, a quick topic search can be carried out over a number of documents indexed according to their key-concepts, which is more precise and efficient than full-text search. Also, key-concepts can also be used to calculate semantic similarity between documents and to cluster the texts according to such similarity (Ricca *et al.*, 2004). Furthermore, key-concepts provide a sort of quick summary of a document, thus they can be used as an intermediate step in *extractive* summarisation in order to identify the text segments that reflect the content of a document. (Jones *et al.*, 2002), for example, exploit key-concepts to rank the sentences in a document by relevance in that they count the number of key-concept stems occurring in each sentence. In the light of the increasing importance of key-concepts in different applications, from search engines to digital libraries, a recent task for the evaluation of key-concept quality was also proposed at the last SemEval-2010 campaign (Kim *et al.*, 2010)

In this work, we assume that the list of n-topmost relevant key-concepts of a text can be used as a sort of summary of the document. We will call this list a *key-concept summary*. We make the hypothesis that, in order to select the abstract that corresponds to an article, we need to find the abstract that is most similar to the key-concept summary extracted from such article. This is achieved by applying a *similarity function* that tries to measure the degree of coherence and the completeness between the key-concept summary and the abstract.

The paper is structured as follows : in Section 2 we provide an overview of the system and briefly describe the four main steps it includes. In Section 3 we detail the architecture of KX, the key-concept extraction tool, providing an insight into its parameter configuration. In Section 4 we present the similarity function applied to weight each possible article/abstract combination, while in Section 5 we describe the algorithm devised for finding the best article/abstract pairs. In Section 6 we detail the experimental setup and the workflow optimization based on training data. Besides, we describe the final setup employed in the test phase and we comment on the evaluation results. Finally, we draw some conclusions and discuss some future improvements of our approach in Section 7.

2 General system description

The algorithm for abstract - article matching that we present includes four steps, as illustrated in Figure 1.

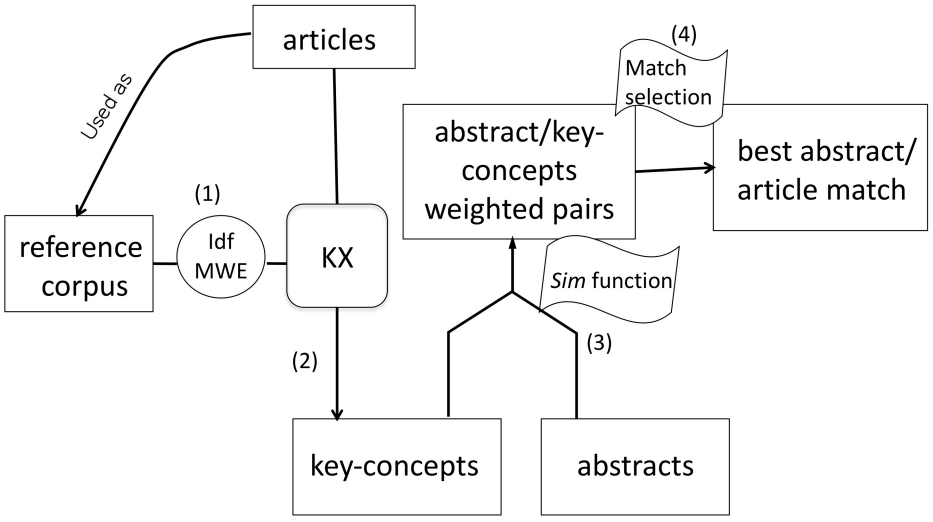

FIGURE 1 – Abstract/article matching workflow

KX (Pianta & Tonelli, 2010), the key-concept extraction component, is involved in the two initial steps. The *first*

one is optional and concerns the extraction of relevant statistical information from a reference corpus. In particular, the tool extracts information about plausible multiword expressions (MWEs) from such corpus and calculates the inverse document frequency of the key-concepts found in each document of the corpus. In the DEFT competition, the reference corpus adopted includes 996 documents, i.e. 300 articles and 300 abstracts from the training set, and 198 articles and 198 abstracts from the test set.

In the *second step*, KX is run over each article to be paired with an abstract in order to extract from each article the corresponding key-concept summary. Details about KX parameter configuration are given in Section 3. Note that KX is an unsupervided system, so the training set is only used to find the best parameter combination for the task and to tune the system for handling French data, but no information about the corresponding abstracts is taken into account.

In a *third step*, for each key-concept summary representing an article and each abstract in the data set a similarity score is computed. In this way, we obtain a complete weighted bipartite graph where the two vertex sets correspond to the articles and to the abstracts, and each article is connected to each abstract by an edge with a weight ≥ 0, assigned by the similarity function.

In the *last* step, we run an algorithm that, given the complete bipartite graph, finds the subgraph corresponding to the best possible match between articles and abstract, in which each vertex can be connected only to another vertex by a single edge.

3 KX : a flexible Key-concepts eXtractor

In this section, we thoroughly describe the basic KX architecture for unsupervised key-concept extraction. With KX, the identification of key-concepts can be accomplished with or without the help of a reference corpus, from which some statistical measures are computed in an unsupervised way. KX is distributed with the TextPro NLP Suite (Pianta *et al.*, 2008) and the current version can handle English and Italian texts. Nevertheless, since the only language-dependent part of the system is the morphological analyzer, we developed for the DEFT competition a new version for French documents, where the morphological analysis is disabled and is replaced by an extensive use of black lists. For details, see the following description.

3.1 Key-concept extraction at document level

Figure 2 displays the key-concept extraction process that, starting from a document, outputs a list of key-concepts ranked by relevance. The same workflow applies both to the extraction of key-concepts from a single document and to the extraction of relevant statistical information about multiwords and key-concepts from a *reference corpus*, which can be optionally used as additional informaton when processing a single document. For more information, see below and in Section 3.2.

The system takes a document in input and first performs tokenization. Then, all possible n-grams composed by any token sequence are extracted, for instance ‘éclipse de soleil’, ‘tous les’, ‘où chacun’. The max length of the selected n-grams can be set by the user and for DEFT it was set to four.

Then, from the n-gram list a sublist of *multiword expressions (MWE)* is derived, i.e. combinations of words expressing a unitary concept, for example ‘compréhension de concepts’ or ‘expression métaphorique’.

In the selection step, the user can choose to rely only on local (document) evidence or to make use also of global (corpus) evidence. As for the first case, a frequency threshold called $MinDoc$ can be set, which corresponds to the minimum number of occurrences of n-grams in the current document. If a reference corpus is also available, another threshold can be added, $MinCorpus$, which corresponds to the minimum number of occurrences of an n-gram in the corpus. KX marks an n-gram in a document as a multiword term if it occurs at least $MinCorpus$ times in the corpus or at least $MinDoc$ times in the document. The two parameters depend on the size of the reference corpus and the document respectively. In our case, the corpus was the set of documents and abstracts made available in the DEFT competition. More details about the thresholds applied can be found in Section 6.

A similar, frequency-based, strategy is used to solve ambiguities in how sequences of contiguous multiwords should be segmented. For instance, given the sequence ‘retour des bonnes manières’ we need to decide whether we

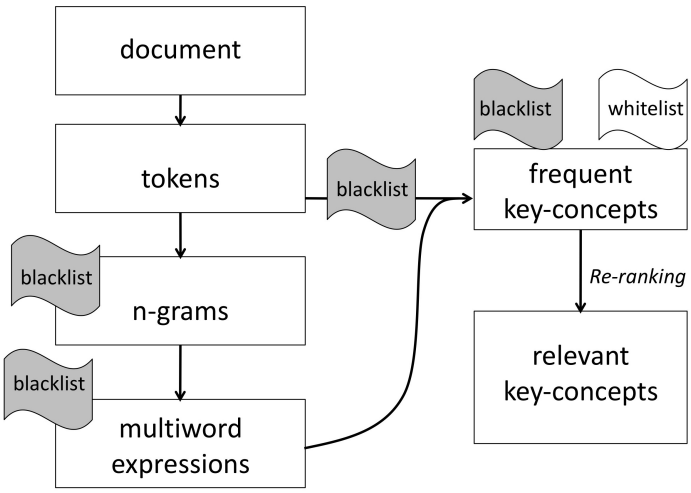

FIGURE 2 – key-concept extraction process

recognize 'retour des bonnes' or 'bonnes manières'. To this purpose, we calculate the strength of each alternative MWE as

$$Strength_{colloc} = docFrequency * corpusFrequency$$

and then choose the stronger one.

In the next step, the single words and the MWEs are ranked by frequency in order to obtain a first list of key-concepts. Thus, frequency is the baseline ranking parameter, based on the assumption that important concepts are mentioned more frequently than less important ones. Frequency is normalized by dividing the number of key-concept occurrencies by the total number of tokens in the current document.

Note that the first key-concept list is obtained by applying *black and white lists* almost at every step of the process, as shown in Fig. 2. This is particularly relevant to the DEFT task, in which we have used a system configuration that does not include the morphological filter usually employed to select the linguistic patterns of MWEs. A black list is applied for discarding n-grams containing one of the language-specific stopwords defined by the user, for example 'avons', 'peut', 'puis', 'parce'. Also single words corresponding to stopwords are discarded when the most frequent tokens are included into the first key-concept list. For example, in French we may want to exclude all key-concepts containing the word 'toi', 'très', 'finalement', etc.

A black list is applied also when selecting MWEs from n-grams. The list can be enriched after running the first system tests, in order to eliminate multiwords that have been selected by mistake. For example, it may be useful to discard n-grams such as 'de plus en plus', 'par exemple', 'en effet', etc.

Finally, black and white lists can be manually compiled also for key-concepts, in order to define expressions that should never be selected as relevant key-concepts as well as terms that should always be included in the key-concept rank. For example, the preposition 'de' is not included in the stopword list because key-concepts like 'problème de reformulation' should be admitted. However, 'de' is very frequent in documents, so it can happen that it is selected as single-word key-concept. In order to avoid this, 'de' can be included in the key-concept black list.

3.2 Parameters for first key-concept ranking

After the creation of a frequency-based list of key-concepts, various techniques are used to re-rank it according to relevance. If a reference corpus is available, as in our case, additional information can be used to understand which key-concepts are more specific to a document, and therefore are more likely to be relevant for such document.

In order to find the best ranking mechanism, and to tailor it to the type of key-concepts we want to extract, the following parameters can be set :

Key-concept IDF : This parameter takes into account the fact that, given a data collection, a concept that is mentioned in many documents is less relevant to our task than a concept occurring in few documents. In order to activate it, a reference corpus must undergo a pre-processing step in which the key-concepts are extracted from each document in the corpus and the corresponding inverse document frequency (IDF) is computed following the standard formula :

$$IDF_k = log\frac{N}{DF_k}$$

where N is the total number of documents and DF_k is the number of documents in the corpus that contain the key-concept k. The IDF of a rare term tends to be high, while the IDF of a frequent one is likely to be low. Therefore, IDF may be a good indicator for distinguishing between common, generic terms and specific ones, which are good candidates for being a key-concept.

When this parameter is activated, for each key-concept found in the current document, its IDF computed over the reference corpus is retrieved and multiplied by the key-concept frequency at document level.

Key-concept length : Number of tokens in a key-concept. Concepts expressed by longer phrases are expected to be more specific, and thus more informative. When this parameter is activated, frequency is multiplied by the key-concept length. For example, if 'expression verbale' has frequency 6 and 'expression verbale des émotions' has frequency 5, the activation of the key-concept length parameter gives 'expression verbale' = 6 * 2 = 12 and 'expression verbale des émotions' = 5 * 4 = 20. In this way, the 4-gram is assigned a higher ranking than the 2-gram.

Position of first occurrence : Important concepts are expected to be mentioned before less relevant ones. If the parameter is activated, the frequency score will be multiplied by the $PosFact$ factor computed as :

$$PosFact = \left(\frac{DistFromEnd}{MaxIndex}\right)^2$$

where $MaxIndex$ is the length of the current document and $DistFromEnd$ is $MaxIndex$ minus the position of the first key-concept occurrence in the text.

The parameters can be independently activated by the user in a configuration file. The key-concept relevance is then calculated by multiplying the normalized frequency of a key-concept by the score obtained by each active parameter. We eventually obtain a ranking of key-concept ordered by relevance. The user can also set the number of top ranked key-concepts to consider as best candidates.

3.3 Parameters for final ranking

After the first ranking described in Section 3.2, a further set of operations can be optionally carried out by the system in order to adjust the preliminary ranking. Again, such operations can be independently activated through a separate configuration file. The parameters have been introduced to deal with so-called *nested* key-concepts (Frantzi *et al.*, 2000), i.e. those that appear within other longer candidate key-concepts. After the first ranking, which is still influenced by the key-concept frequency, *nested* (shorter) key-concepts tend to have a higher ranking than the containing (longer) ones, because the former are usually more frequent than the latter. However, in some settings, for example in scientific articles, longer key-concepts are generally preferred over shorter ones because they are more informative and specific. In such cases, the user may want to adjust the ranking in order to give preference to longer key-concepts and to reduce or set to zero the score of nested key-concepts.These operations are allowed by activating the following parameters :

Shorter concept subsumption : It happens that two concepts can occur in the key-concept list, such that one is a specification of the other. Concept *subsumption* and *boosting* (see below) are used to merge or rerank such couples of concepts. If a key-concept is (stringwise) included in a longer key-concept with a higher frequency-based score, the score of the shorter key-concept is transferred to the count of the longer one.

For example, if 'expression verbale' has frequency 4 and 'expression verbale des émotions' has frequency 6, by activating this parameter the relevance of 'expression verbale des émotions' is 6 + 4 = 10, while the relevance of 'expression verbale' is set to zero. The idea behind this strategy is that nested key-concepts can be deleted from the final key-concept list without losing relevant information, since their meaning is nevertheless contained in the longer key-concepts.

Longer concept boosting : This parameter applies in case a key-concept is (stringwise) included in a longer key-concept with a lower relevance. Its activation should better balance the ranking in order to take into account that longer n-grams are generally less frequent, but not less relevant, than shorter ones. The parameter is available in two different versions, having different criteria for computing such boosting. With the *first option*, the average score between the two key-concepts relevance is computed. Such score is assigned to the less frequent key-concepts and subtracted from the frequency score of the higher ranked one. With the *second option*, the longer key-concepts is assigned the frequency of the shorter one. In none of the two variants key-concepts is deleted from the relevance list, as it happens by activating the *Shorter concept subsumption* parameter.

For example, if 'expression verbale' has score 6 and 'expression verbale des émotions' has score 4, by activating the first option of this parameter the relevance of 'expression verbale' becomes 6 - ((6 + 4) / 2) = 1, while the relevance of 'expression verbale des émotions' is set to 5, i.e. (6 + 4) / 2 .

With the second option, both the relevance of 'expression verbale des émotions' and of 'expression verbale' is set to 6.

As shown in the examples above, these parameters can change the output of the ranking by deleting some entries and boosting some others. Note that after applying one cycle of subsumption/boosting, the order of the concepts can dramatically change, producing the conditions for further subsumption/boosting of concepts.

The number of iterations for the application of this re-ranking mechanism can be set by the user, and each cycle increases the impact of the re-ranking on the key-concept list. The parameters can be activated together and in different combinations. If all parameters are set, the short concept subsumption procedure is applied first, then the longer concept boosting is run on the output of the first re-ranking, so that the initial relevance-based list goes through two reordering steps.

4 Similarity score assignment

After a key-concept summary has been extracted from each document, we compute a similarity score between such summary and each candidate abstract. The final goal is to match the document/abstract pair achieving the highest similarity. In particular, we believe that the similarity between a key-concept summary and an abstract can be measured in terms of *coherence*, i.e. the amount of information in the key-concept summary that is present also in the abstract, and *completeness*, i.e. the amount of information in the abstract that is represented through the key-concept summary. In other words, the key-concept summary should ideally contain *all and only* the information in the abstract.

The coherence and completeness criteria can be cast in terms of *precision* and *recall* of the key-concepts extracted from a text, taking a manual abstract as gold standard of the extraction task. This approach is inspired by the methodology originally proposed by (Pianta, 2011) for the *PatExpert* system [1] to evaluate the quality of the key-concepts extracted from a patent document. Since every patent in the PatExpert collection was provided with an abstract, the author proposed to compute the *precision* of a key-concept summary as the percentage of the top-ranked key-concepts that are mentioned in the abstract, and *recall* as the percentage of words in the abstract that appear among the top-ranked key-concepts. Then, F1 was additionally computed following the standard formula :

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

We rely on the same assumption that high precision, recall and F1 correspond to a high similarity between key-concept summary and abstract, but apply the metrics as a similarity score instead of as an evaluation metric.

1. http://www.patexpert.org/

Therefore, we refer to them as Sim_{Prec}, Sim_{Rec} and Sim_{F_1}. Following this idea, we consider Sim_{Prec} as a measure of coherence, and we define it as :

The percentage of n-grams in the key-concept summary that are contained also in the abstract.

On the other hand, we see Sim_{Rec} as a measure of completeness and we define it as :

The percentage of tokens in the abstract, excluding stopwords, that occur also in the key-concept summary.

After carrying out some tests on the DEFT training set, we assess that Sim_{F_1} is the best similarity score to use to our purpose, while Sim_{Prec} and Sim_{Rec} alone do not achieve comparably satisfactory results.

Sim_{Prec} and Sim_{Rec} are calculated according to the procedure described in Algorithm 1 :

Algorithm 1 Compute Sim_{Prec} and Sim_{Rec} for Sim_{F_1} assignment

Require: Key-concept summary K and abstract A
  initialise $count = 0$
  for all $k \in K$ **do**
    if k appears in A **then**
      $count \Leftarrow count + 1$
      remove all occurrences of k in A
    else
      repeat
        take longest sub-phrase $sub \in k$
      until
      sub appears in A
      $count \Leftarrow count + \frac{subLength}{kLength}$
      remove all occurrences of sub in A
    end if
  end for
  $Sim_{Prec} = \frac{count}{|K|}$
  remove stopwords from A
  $Sim_{Rec} = \frac{AInitialLength - AFinalLength}{AInitialLength}$

Given the set of key-concepts K extracted from an article and an abstract A, we compare them as follows : if $k \in K$ matches a phrase in A, than we remove all occurrences of such phrase from A. Otherwise we look for the longest subphrase of k matching a phrase in A, and again if a match is found, all matching phrases are removed. If we can match a full key-concept this is counted as 1 score. If we can only match a subphrase, we count a fraction of 1 depending on the length of the matching sub-phrase. For instance if the full key-concept includes 3 tokens and we can only match a subphrase of length 2, the score is calculated as 2/3.

Once this match-and-delete procedure is carried out for the top-ranked key-concepts, we calculate the percentage of $k \in K$ that are included in A, which corresponds to Sim_{Prec} between K and A. Then, we take the words that are left after the deletion of matching key-concepts and subphrases, we further delete any stopword, and we compare the length in words of the abstract before and after the match-and-delete procedure (i.e. $AFinalLength$ vs. $AInitialLength$). The number of deleted words ($AInitialLength - AFinalLength$) divided by the number of tokens in the abstract ($AInitialLength$) excluding stopwords gives an estimation of the concepts in the abstract that have been actually extracted by KX. In short, it measures the recall of the algorithm. In this way, we calculate also the Sim_{Rec} for each K - A pair. Finally, we use the two measures to compute Sim_{F_1}.

5 Algorithms for best match selection

After assigning a similarity score to each possible pair of key-concept summary and abstract, we need to decide which are the best document - abstract matches. Since the output of the previous step is composed of two partitions,

the documents D and the abstract collection AC, where each source node $d \in D$ is connected to each target node $a \in AC$ and vice versa by a weighted link, we consider the pair selection task as a matching problem in a bipartite graph.

More formally, a weighted bipartite graph is a graph $G = (V, E)$ where V is the node set partitioned into two nonempty sets V_1 and V_2, and every node edge $e \in E$ connects a node in V_1 to a node in V_2 with a weight. In our task, V_1 and V_2 correspond respectively to the set of documents D and the set of abstracts AC. Besides, the graph G is *complete*, i.e. all nodes in D and in AC are connected to each other by a weight ≥ 0 computed as described in Section 4. Such weight expresses the similarity between a document and an abstract.

The initial configuration of the complete graph is riported in Figure 3 (left graph). Our goal is to reduce the graph to a perfect matching subgraph, where each $d \in D$ is mapped to one $a \in AC$ and vice versa (Fig. 3, right graph).

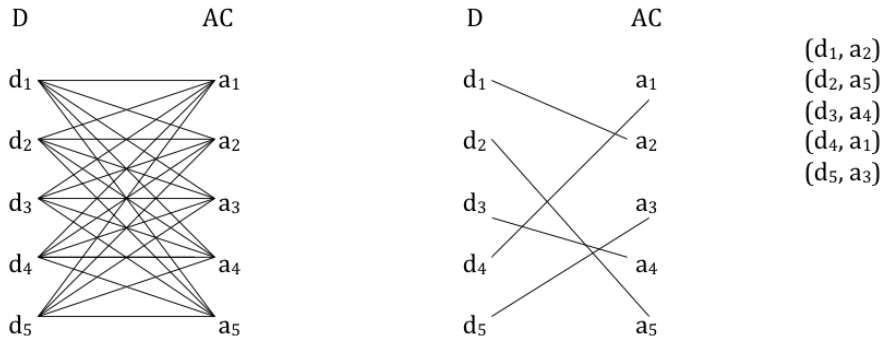

FIGURE 3 – From a complete bipartite graph to a perfect matching between (d_i, a_i) pairs

In order to select the best (d_i, a_i) pairs, we test two different matching algorithms : one is the so-called *Hungarian Algorithm* (Kuhn, 1955), a classical combinatorial optimization algorithm used for maximum assignment problems [2]. This algorithm is exploited when, given a complete weighted bipartite graph, it is necessary to find a perfect matching with maximum cost, i.e. the subgraph that maximizes the total weight of the matches. The intuition behind it is to find the set of edges that at *global* level achieves the highest weight.

We also test a second approach, which we call *Local Match* (*LocMatch*), aimed at finding the set of edges that achieves the highest weight at *local* level. In short, we start the match by choosing the (d_i, a_i) pair having the highest weight in the graph, and we iteratively repeat the assignment by choosing at each stage the edge with the maximal increase of the weight.

In order to explain the different behaviour of the two algorithms, we report an example in Fig. 4. Given a set of sample documents $\{d_1, d_2, d_3\}$ and a set of abstracts $\{a_1, a_2, a_3\}$ connected by weighted edges, we can represent this graph through a rectangular matrix, where the weight associated to each link is reported. The matches assigned by the algorithm are highlighted in grey. With the *Hungarian Algorithm* we obtain the perfect match represented in the central matrix of Fig. 4. In this case, the algorithm finds all the edges with the optimal total weight, i.e. $8 + 4 + 5 = 17$.

With the second algorithm (matrix on the right), we start from matching the (d_i, a_i) pair with the highest weight, i.e. d_1 and a_3. Consequently, we delete the third columns and the first row of the matrix, because d_1 and a_3 cannot be paired with other candidates. Then, we match the pair with the highest weight in the remaining matrix, i.e. d_2 and a_1. Finally, the only possible pairing left is between d_3 and a_2. In this case, the total match weight is $8 + 6 + 2 = 16$, which is lower than the weight obtained with the other algorithm. However, more relevance is given to weights at *local* level.

2. We use the Perl implementation available at `http://search.cpan.org/dist/Algorithm-Munkres/`

	a_1	a_2	a_3
d_1	3	0	8
d_2	6	4	3
d_3	5	2	7

	a_1	a_2	a_3
d_1	3	0	8
d_2	6	4	3
d_3	5	2	7

	a_1	a_2	a_3
d_1	3	0	8
d_2	6	4	3
d_3	5	2	7

FIGURE 4 – Matrix representing the initial complete bipartite graph (left), assignments done with the *Hungarian algorithm* (middle) and with the second matching algorithm (right).

6 Experimental Setup and Evaluation

In the DEFT task for “Abstract/article matching", two subtasks are proposed : the first one consists in identifying the best matches between abstracts and full scientific articles, while in the second subtask the matches must be found between abstracts and articles where introduction and conclusions have been removed.

The training corpus comprises 300 abstracts and 300 articles (full and shortened) from 5 different journals, explicitly indicated for each abstract and each article. The test corpus comprises 198 abstracts and 198 articles from 6 journals, i.e. those already used in the training corpus plus a new one.

Since our matching system does not require supervised learning, we use the training set as a development set, to perform tests and tune the required parameters.

6.1 Training phase

We report in Table 1 the best parameter combination with the corresponding results obtained on the training set. For details about KX parameters, see the explanation in Section 3.

	Subtask 1 : Full articles	Subtask 2 : Short articles
KX Parameters		
N. of key-concepts extracted from each article	60	30
$MinCorpus$	8	8
$MinDoc$	2	2
Use $corpusIdf$	No	Yes
Multiply relevance by key-concept length	Yes	No
Consider position of first occurrence	Yes	No
Shorter concept subsumption	Yes (1 cycle)	Yes (1 cycle)
Longer concept boosting (assign frequency of shorter key-concepts to longer ones)	Yes (1 cycle) Yes	Yes (1 cycle) Yes
Similarity function for similarity score assignment	Sim_{F_1}	Sim_{F_1}
Graph matching algorithm	$LocMatch$	$LocMatch$
Journal-based match	No	No
P, R, F1 on training set	P 0.976, R 0.966, **F1 0.971**	P 0.943, R 0.940, **F1 0.941**

TABLE 1 – Best parameter combination for training set

As expected, the final results obtained using the full articles outperform those based on the articles without introduction and conclusions. This shows that the information at the beginning and at the end of an article is usually crucial to the identification of the most relevant concepts of the document.

Another difference is the number of top-ranked key-concepts considered for the match : in shorter articles 30 terms are enough, while in full ones 60 key-concepts seem to capture better the document content. Besides, in the classification of shorter articles, the integration of $corpusIdf$ information (computed over all training documents)

in the ranking process improves on the matching performance, while for full articles is does not provide additional useful information. As expected, in the second subtask the information about the position of the first occurrence of the key-concept does not help, because the introduction has been removed. Also, longer key-concepts are more relevant to the match of full articles than of shortened ones (see parameter *Multiply relevance by key-concept length*).

As for the similarity function, we tested also Sim_{Prec} only and Sim_{Rec} only, but the best results are achieved using their weighted average Sim_{F_1}.

We also experimented with both graph matching algorithms, but the $LocMatch$ always outperforms the *Hungarian Algorithm* (best F1 is 0.68 on full articles and 0.69 on shortened articles).

Finally, we tested two different experimental settings, one including all documents in a single corpus, and one subdividing them into smaller corpora according to the corresponding journal, and performing the abstract - article match only among documents coming from the same journal. For example, 60 articles and 60 abstracts in the training set have been extracted from the journal "Anthropologie et Sociétés", so we have evaluated also the performance of our system on this subclass of documents (see parameter *Journal-based match*). Surprisingly, this kind of approach does not improve on the system performance on the training set, because we achieved as best F1 0.965 on full articles.

6.2 Test phase

In the test phase, we submitted three system runs on the first subtask and three runs on the second one. For both tasks, one run was based on the parameter configuration in Table 1, a second run was computed by considering also the $corpusIdf$ (this time computed over all training and test documents), and a third one by including the $corpusIdf$ in a journal-based setting. Evaluation results are displayed in Table 2. Note that this time we report only one accuracy score obtained with the official task scorer, which corresponds to the percentage of correct matches.

	Subtask 1 : Full articles	Subtask 2 : Short articles
Setting as in Table 1	0.960	0.934
+ $corpusIdf$	0.975	**0.964**
+ journal-based match	**0.990**	**0.964**

TABLE 2 – System evaluation on test set

The performance on the test set outperforms the results obtained on the training set, probably because of the smaller amount of documents to match. The $corpusIdf$ is now relevant to the matching quality because it is computed on a larger corpus, comprising 996 documents (498 full articles and 498 abstracts), while in the training phase only 600 documents were available. This means that the idf information is important in the matching task only if it is computed on a large corpus. Also, the journal-based setting on the test set improves on the system performance with full articles because it significantly reduces the number of possible matches. However, the improvement is not achieved on shortened articles, probably because the content of the documents coming from the same journal tends to be more homogeneous and more difficult to discriminate if information-rich parts such as introduction and conclusions are removed.

7 Conclusions

In this paper we have presented a system for abstract - article match. It first extracts a list of key-concepts from each document, then a similarity score is applied between the key-concept lists and the abstracts, and finally a graph match algorithm is applied in order to identify the best matches.

The key-concept extraction component was created as a standalone system and was previously evaluated in the SemEval-2010 campaign on English texts with good results. For DEFT, we developed an extension of the system

which is able to handle French texts without performing any syntactic analysis. Besides, the system does not need a large traning corpus because only a small development set for parameter tuning is required.

The system performed well also in this case, since it was able to correctly match almost all pairs in the test set. In the future, we plan to extend the key-concept extraction component also to new languages, and we are currently working at the integration of the French morphological analyzer *Morfette* (Chrupala *et al.*, 2008) into the extraction pipeline.

Acknowledgements

This work has been partially funded by the European Commission under the contract number FP7-248594, PESCaDO project [3]. We thank Elena Cabrio for the support with the French language.

Références

CHRUPALA G., DINU G. & VAN GENABITH J. (2008). Learning Morphology with Morfette. In *Proceedings of the 6th International Conference on Languages Resources and Evaluations (LREC 2008)*, Marrakech, Morocco.

FRANTZI K., ANANIADOU S. & MIMA H. (2000). Automatic recognition of multi-word terms : the C-value/NC-value. *Journal of Digital Libraries*, **3**(2), 115–130.

JONES S., LUNDY S. & PAYNTER G. (2002). Interactive Document Summarisation Using Automatically Extracted Keyphrases. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, Hawaii.

KIM S. N., MEDELYAN O., KAN M.-Y. & BALDWIN T. (2010). SemEval-2010 Task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of SemEval 2010, Task 5 : Keyword extraction from Scientific Articles*, Uppsala, Sweden.

KUHN H. W. (1955). The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, **2**, 83–97.

PIANTA E. (2011). Content Distillation from Patent Material. Draft of a Book Chapter on the PatExpert system (to appear).

PIANTA E., GIRARDI C. & ZANOLI R. (2008). The TextPro tool suite. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC)*, Marrakech, Morocco.

PIANTA E. & TONELLI S. (2010). KX : A flexible system for Keyphrase eXtraction. In *Proceedings of SemEval 2010, Task 5 : Keyword extraction from Scientific Articles*, Uppsala, Sweden.

RICCA F., TONELLA P., GIRARDI C. & PIANTA E. (2004). An empirical study on keyword-based web site clustering. In *Proceedings of the 12th IWPC*, Bari, Italy.

3. http://www.pescado-project.eu/