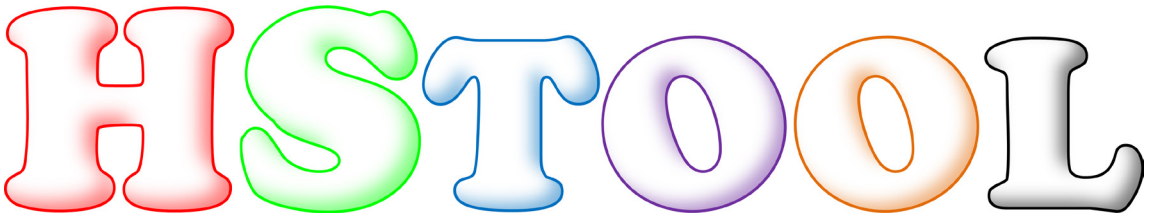
FOI

HSTOOL for Horizon Scanning of Scientific Literature

MAJA KARASALO, JOHAN SCHUBERT



FOI-R--4760--SE
ISSN 1650-1942 April 2019

Maja Karasalo, Johan Schubert

HSTOOL for Horizon Scanning of Scientific Literature

Bild/Cover: Johan Schubert

Titel	HSTOOL for Horizon Scanning of Scientific Literature
Title	HSTOOL för avskanning av vetenskaplig litteratur
Rapportnr/Report no	FOI-R--4760--SE
Månad/Month	April
Utgivningsår/Year	2019
Sidor/Pages	35 p
Kund/Customer	Swedish Armed Forces
Forskningsområde	12. Övrigt
FoT-område	Ej FoT
Projektnr/Project no	E72790
Godkänd av/Approved by	Christian Jönsson
Ansvarig avdelning	Ledningssystem

Sammanfattning

Denna rapport beskriver en metodik och ett system för så kallad horizon scanning av vetenskaplig litteratur i syfte att upptäcka vetenskapliga trender. Litteratur inom ett brett definierat forskningsfält kan med denna metodik grupperas automatiskt i kluster efter ämnesinnehåll och rangordnas med avseende på inflytande inom respektive ämnesområde. En metod för att bestämma det optimala antalet kluster för en befintlig dokumentklustringsalgoritm, samt en metod för att ta fram beskrivande och särskiljande ord för de upptäckta klustren introduceras. Dessutom föreslås en rankingmetodik baserad på citeringsstatistik för att identifiera inflytelserika bidrag inom de upptäckta ämnesområdena.

Nyckelord: horizon scanning, scientometri, Gibbs sampling, Dirichlet multinomial mixture model, entropi, klustring, HSTOOL

Summary

In this report we develop a methodology and a system for horizon scanning of scientific literature to discover scientific trends. Literature within a broadly defined field is automatically clustered and ranked based on topic and scientific impact, respectively. A method for determining the optimal number of clusters for the established Gibbs sampling Dirichlet multinomial mixture model (GSDMM) algorithm is proposed along with a method for deriving descriptive and distinctive words for the discovered clusters. Furthermore, we propose a ranking methodology based on citation statistics to identify significant contributions within the discovered subject areas.

Keywords: horizon scanning, scientometrics, Gibbs sampling, Dirichlet multinomial mixture model, entropy, clustering, HSTOOL

Contents

1 Introduction

Horizon scanning methods aim to discover changes, disruptions, and trends with the potential to influence the development of a particular area of interest significantly. For scientific literature, the goal of horizon scanning is to discover emerging or rapidly growing research areas and to identify technologies that have reached a level of readiness that is suitable for industrial applications.

To scan broad scientific fields without making presumptions about specific topics worthy of further studies, large numbers of scientific articles must be included in the scanning process. This requirement motivates the need for a semiautomatic approach, where software tools provide some initial filtering and structuring of the data.

In this report, we propose a method for semiautomatic horizon scanning of scientific literature and present the horizon scanning system HSTOOL that supports the proposed method. The goal of the method is to identify rapidly developing fields and their most significant contributions by first scanning the scientific literature using relatively general search criteria and then structuring and filtering the discovered articles. HSTOOL accesses the Thomson Reuters Web of Science[1] (WOS) Core Collection through a set of APIs that allow searches and retrieval of article data, as well as citation statistics. The proposed method and software thereby enable semiautomatic scanning of 71 million articles found in over 20 300 journals, 94 000 books and 180 000 conference proceedings included in the WOS Core Collection.

The key steps of the method are clustering of the discovered literature to identify topics and ranking of articles in the resulting clusters based on scientific citation statistics to find the most significant contributions within the respective topic.

We use the Gibbs sampling Dirichlet multinomial mixture model (GSDMM) algorithm [1] for clustering and introduce a complementary method to determine the optimal number of clusters. We find the optimal clustering by evaluating the quality of placement of every article in each specific cluster using an entropy measure [2], [3]. Furthermore, we develop a method for automatically presenting two sets of descriptive words for each cluster based on the cluster's contents. The first set consists of the words that most often occur in the cluster, while the second set consists of the most distinctive words in the sense that their occurrence throughout the entire set of articles is concentrated in the current cluster. In combination, the sets provide a description of the articles that are part of the cluster and an account of what primarily distinguishes these articles from articles in other clusters.

For scientific ranking, we propose a set of scientometric measures that identify articles that have made a significant impact in the respective fields. Influence is measured as either collecting many citations over a short period of time, or having a strong citation trend, or frequently being cited in prestigious journals. Finally, the measures are aggregated into a total ranking within each discovered cluster. The top-ranked articles can thus be selected for detailed study.

The report is organized as follows. In Section 2, we describe a workflow model that contains all process steps of searching, organizing and analyzing scientific articles. In Section 3, we develop methods for performing horizon scanning to discover and analyze trends in scientific literature. In Section 4, we develop processes for scientific trend discovery and describe a literature scanning system. We apply the system to a case study of literature on military applications of artificial intelligence (Section 5). Finally, conclusions are provided in Section 6.

[1] http://www.webofknowledge.com (March 2019).

2 Workflow

Fig. 1 shows the proposed workflow of horizon scanning of scientific literature in five steps. The process is intended to facilitate scanning of broad areas defined by a general topic search string (step 1). The topic search is further discussed in Section 3.1. Once a search has been performed and records downloaded (step 2), topics are automatically discovered using a clustering algorithm that groups the scientific articles based upon textual contents (step 3). Details of the clustering algorithm are described in Sections 3.2 and 3.3. Clusters of articles can then be selected for further studies. To find the key contributions from a cluster of interest, a ranking method is proposed that uses a set of statistical citation measurements to capture various aspects of scientific impact (step 4). Section 3.4 covers the derivation of impact measurements and ranking procedures. Once top-rated contributions for a subject area have been identified, a manageable subset of articles can be selected for detailed studies (step 5).

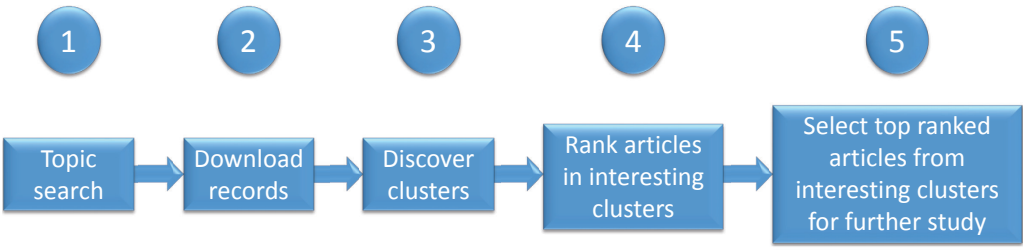


Figure 1. Proposed workflow for horizon scanning of scientific literature.

3 Methodology

In this section, we describe methods for searching scientific literature, clustering articles in groups that correspond to subject areas and evaluating the scientific impact of all articles with citation statistics. Search methods are described in Section 3.1, clustering of articles in Section 3.2, methods for describing the contents of clusters in Section 3.3 and the ranking of articles according to citation statistics in Section 3.4.

3.1 Searching scientific publications

All searches are performed using search terms provided by subject matter experts. These search terms should be tested before use in HSTOOL to ensure that they yield results within the area of interest.

We use HSTOOL to search for publications through an API that provides access to WOS. We limit the search to the Core Collection because that database has the citation statistics that we need for scientometric analysis and ranking of articles.

3.1.1 Search terms

Usually, we search in the topic field (TS) with a search query that consists of the search terms supplied, along with logical operators AND, OR, NOT, and NEAR. In this instance, NEAR*n* allows *n* words between two terms. However, there is nothing to prevent us from searching all available fields with a logical construct of search terms, Boolean operators and parentheses.

3.2 Clustering of articles

Once a search result has been downloaded from WOS (Fig. 1, steps 1–2) we want to group all articles that concern the same subject area into a cluster to be treated as a separate subproblem (Fig. 1, step 3) and then use scientometric information to determine which articles within each cluster are most important to that area (Fig. 1, step 4).

In the following two sections, we describe how to use a GSDMM algorithm to organize articles into clusters with common subject areas (Section 3.2.1) and how we determine the optimal number of clusters (Section 3.2.2). It is important to point out that the search terms used in the previous step are not used in the clustering phase.

3.2.1 Clustering with GSDMM

To group articles within the same subarea, we use the abovementioned GSDMM [1], [4]. Simply described, this method starts from a large number of clusters and a random distribution of articles among clusters. Then, the method examines each article to determine if it fits better in any other cluster than where it is currently placed. This procedure is repeated iteratively for all the articles until there are no more changes.

The method proceeds by comparing for every article all words in the article's title and abstract with the corresponding words in all other articles. If a word is missing or occurs a different number of times when comparing with another article, the probability that these articles belong together is assigned a lower value. These probabilities are combined for all articles within each cluster (and also for the cluster where the article is currently located). This results in an evaluation for all clusters of how well this article fits into all the different clusters. Then, the article is moved to a cluster where it fits well according to these probabilities. The procedure is applied to all articles and repeated iteratively until all articles are placed in their best clusters.

During the process, the number of clusters will decrease dramatically, often by 80–85%. For example, if we start with 500 clusters and thousands of articles, we can finish with 75–100 clusters.

The clustering process is performed by a sequence of Gibbs sampling iterations. During each iteration, we calculate the probability of each article belonging to each cluster *k*, resulting in the probability that the article should be moved to that cluster.

We have [1]

$$p_{dki}\left(k_d = k \middle| \vec{k}_{\neg d}, \vec{d}\right) \propto \frac{m_{k,\neg d} + \alpha}{D - 1 + K\alpha} \times \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} \left(n_{k,\neg d}^w + \beta + j - 1\right)}{\prod_{i=1}^{N_d} \left(n_{k,\neg d} + V\beta + i - 1\right)}, \quad (1)$$

where on the left-hand side, k_d is the cluster position of article *d*, *k* is the *k*th cluster, $\vec{k}_{\neg d}$ is the set of cluster positions of all other articles excluding *d*, and $\vec{d}$ is the set of all articles. In the first term on the right-hand side, $m_{k,\neg d}$ is the number of articles in cluster *k* not including *d*, α is a cluster parameter set to 0.1 in our test case, *D* is the total number of articles under consideration, and *K* is the initial number of clusters. In the second term on the right-hand side, *w* is the *w*th word of article *d*, N_d^w is the number of times word *w* appears in article *d*, $n_{k,\neg d}^w$ is the number of times word *w* appears in cluster *k* when article *d* has been removed, β is a cluster parameter that will determine the number of final clusters, N_d is the number of words in article *d*, $n_{k,\neg d}$ is the number of words in cluster *k* when article *d* has been removed, and *V* is the number of words in the vocabulary.

During the first iteration, a new cluster position is sampled for each article using (1). After each sampling, (1) is updated. When all articles in *D* have been reassigned to new cluster positions, the second iteration starts. The process continues for a fixed number of iterations. The final cluster positions of all articles at the last iteration is the result of the clustering process.

3.2.2 Managing the number of clusters

To select the best number of clusters, we need to evaluate various options. To this end, we evaluate various numbers of clusters based on the quality of clustering.

The GSDMM algorithm does not require a predetermined number of clusters to assign the articles of a given corpus[2]. However, the number of clusters depends on parameter $\beta \in (0, 1)$ that appears in equation (1). A value of β near zero results in many clusters, while β near one produces fewer clusters.

Several standard internal clustering performance metrics [5] utilize some definition of distance between data points. However, since the GSDMM algorithm does not utilize any distance measure between documents to define clusters, these metrics are inapplicable. Instead, we focus on the articles that have been clustered and study how well they fit in the clusters where they have been placed.

Each article has a probability distribution across all clusters that indicates the probability that each cluster is the optimal location for that article (1). This distribution is calculated and used in the clustering process for GSDMM and is recalculated in each step of the clustering process for all articles. At the end of the clustering process, we use the final calculated probability distribution for each article. This is a distribution over all initial

[2] The collection of all articles from a particular search.

clusters, although most of the original clusters are empty at the end of the clustering process and thus have a nearly zero probability.

We consider $\{p_{dki}\}$, where p_{dki} is the probability that article d belongs to cluster k at iteration i (1), with

$$\sum_{k=1}^{K} p_{dki} = 1 \tag{2}$$

for any constant d and i, and where K is the initial number of clusters.

If the placement of a particular article is almost certain, that article will have a probability near one for the respective cluster. Sometimes, an article may have more than one probability that is not near zero because the placement is uncertain. A clustering can be considered to be of high quality if as many articles as possible have as certain a placement as possible. Consequently, the entropy of the probability distribution is a good measure of the quality of placement of a particular article [2]. To study the convergence of the GSDMM algorithm, we calculate at each Gibbs sampling iteration i the entropy for each article d as

$$Ent_{di} = -\sum_{k=1}^{K} p_{dki}\left(k_d = k \middle| \vec{k}_{\neg d}, \vec{d}\right) \log\left[p_{dki}\left(k_d = k \middle| \vec{k}_{\neg d}, \vec{d}\right)\right]. \tag{3}$$

To determine the quality of a specific clustering (i.e., the clustering at a specific iteration i for a specific value of β), we calculate its entropy as

$$Ent_i = \sum_{d=1}^{D} Ent_{di}. \tag{4}$$

Fig. 2 shows the convergence of Ent_i for $i \in [0, 14]$, averaging over 100 runs for a test case. The clustering quality is not significantly improved after 10 iterations. However, entropy convergence can vary between runs, which would motivate a dynamic choice of iterations, whereby the entropy reduction rate determines when the algorithm is complete. This would be an interesting future direction to investigate.

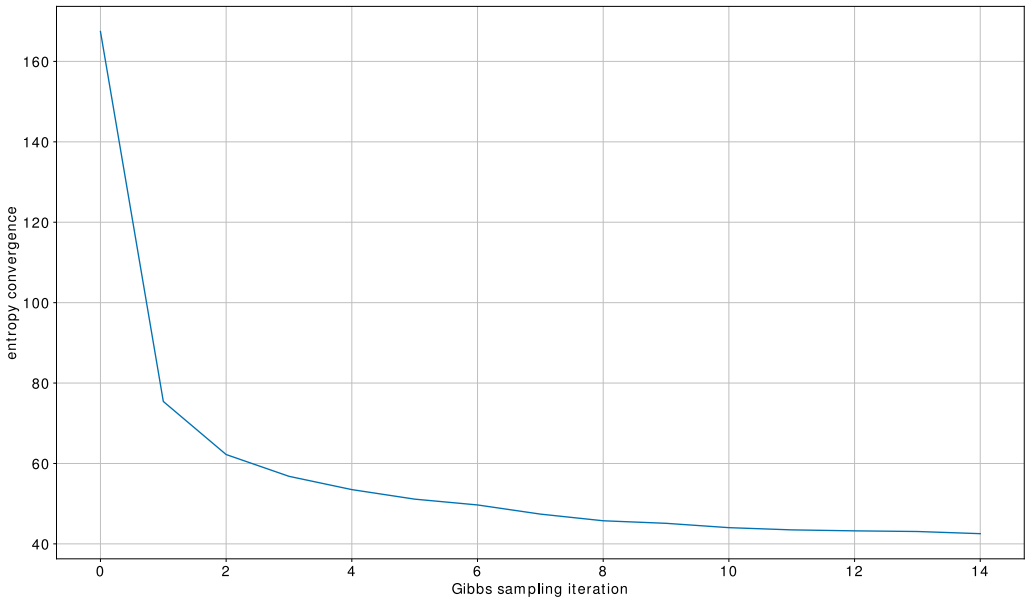


Figure 2. Entropy convergence (4) over 15 Gibbs sampling iterations for all articles in a test case, averaging over 100 runs.

From the above, it follows that a good measure of quality of the entire partition of all articles for a particular clustering process is the sum of entropy over all articles after the final iteration, where Ent_{14} is the sought-after entropy to be minimized.

In Fig. 3, the average number of discovered clusters is shown for a test case for various values of parameter β. If β is small, we obtain a large number of remaining clusters at the end of the process. The number of clusters drops rapidly if β is increased. For values of $\beta > 0.2$, the decrease in the number of final clusters is more gradual. In a test case with $\beta = 0.01$, GSDMM discovers on average 326 clusters among 1358 articles, while for $\beta = 0.99$, the average is six clusters. It is clear that choosing the right value of β is key to obtaining an appropriate number of clusters for the contents of the corpus.

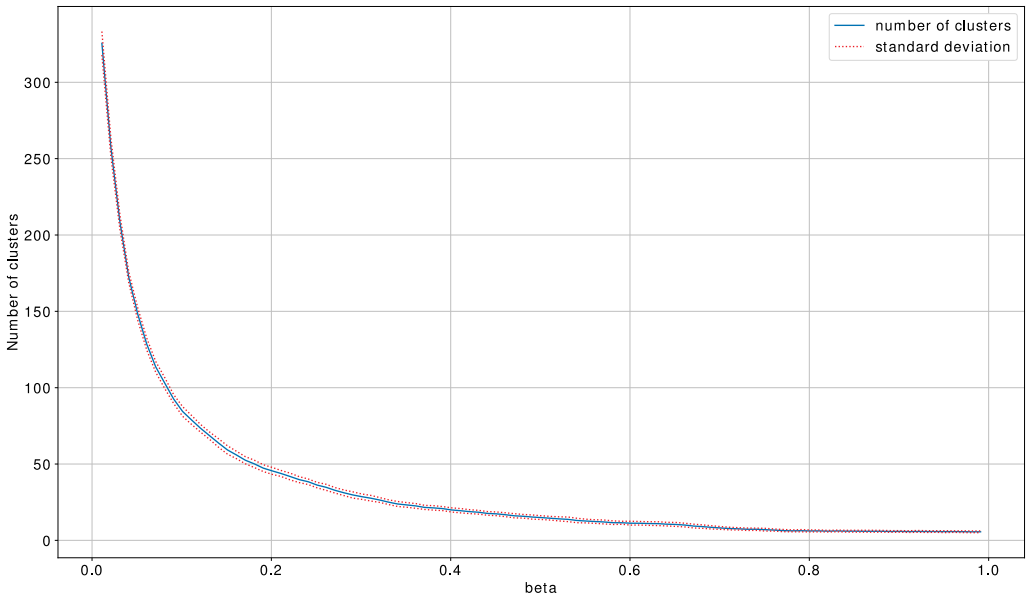


Figure 3. Average number of clusters discovered by GSDMM as a function of β, averaged over 100 runs for each value of β for a test case.

To find the best number of final clusters, we study the entropy at the end of each clustering process for values of β between zero and one. The results are shown in Fig. 4.

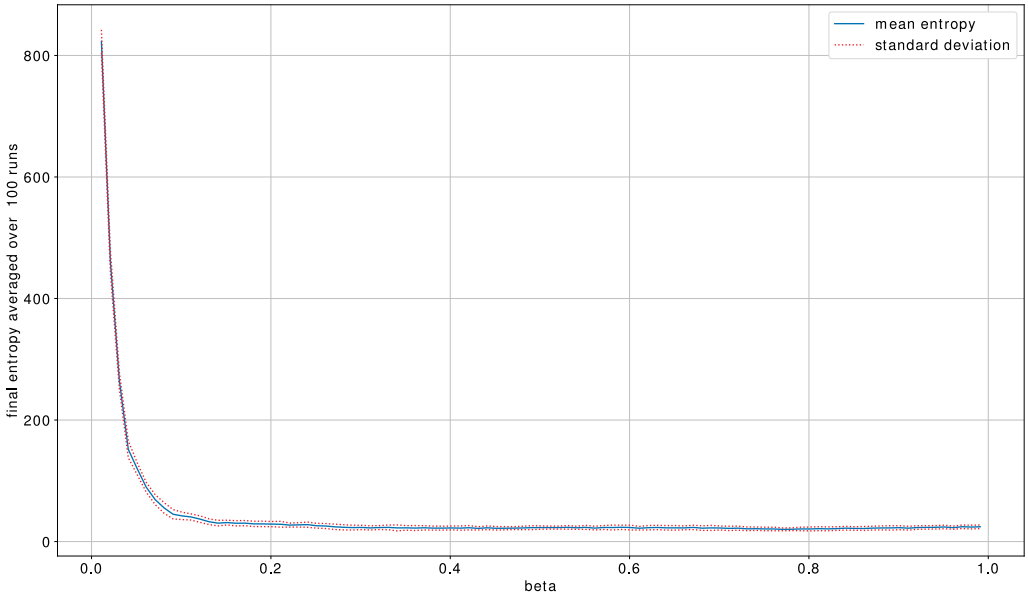


Figure 4. Final entropy Ent_{14} (3) for a test case summed over all articles as a function of β, averaged over 100 runs.

As β increases, there is a decline in the final entropy for each clustering process. Note that most of the decline in entropy occurs when β is increased to 0.1. For values of $\beta > 0.1$, the decline in entropy is modest. The initial high entropy is caused by partitioning the set of articles into too many clusters of overlapping subject areas relative to the current problem.

However, the number of clusters keeps decreasing as β approaches 1, as shown in Fig. 3, without any improvement in entropy (Fig. 4), which ultimately results in a few large clusters, each containing multiple topics. Ideally, we want to find a partition that has well-defined clusters that correspond to subject areas yet has the lowest possible entropy.

To estimate the correct number of clusters, the final entropy derived from clusterings with various values of β is calculated. As shown in Fig. 4, if β is small, entropy is high; as β increases, entropy declines with a small residual entropy at high β. It is evident that there is a change of behavior of the entropy at a point that we consider to yield the best number of clusters; that point is determined as follows [3]. The concave lower envelope of entropy is determined by a convex hull algorithm. At any abscissa, the envelope function is bisected into left and right parts. The acute angle between the left and right line segments is minimized across all bisection values of abscissa, and the minimizing abscissa is selected as the best value of β, see Fig. 5.

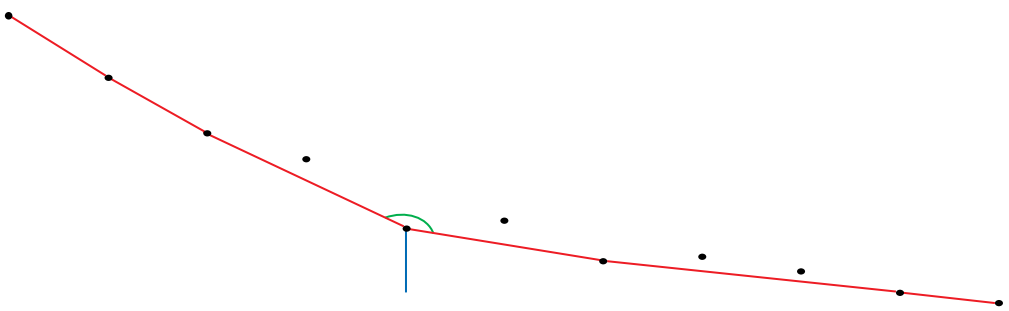


Figure 5. Red line is the concave lower envelope of the black dots. Blue line is an abscissa *a* of point (*a, b*), and green is the minimizing angle.

3.3 Describing the contents of clusters

In this section, we outline a method for describing the contents of a cluster. A high-level description is given by the most representative and the most distinctive words. The most representative words are those that most often occur in the cluster. For cluster *k*, we have

$$F_k^w = n_k^w, \tag{5}$$

where n_k^w is the number of times word *w* occurs in cluster *k*. We rank all words in cluster *k* according to F_k^w and present the highest-ranked words with the maximum F_k^w as representatives of cluster *k*.

Words that distinguish a cluster from other clusters are determined by calculating the entropy of each word in the corpus as

$$E_w = -\sum_{k=1}^{K} \frac{n_k^w}{\sum_{j=1}^{K} n_j^w} \log\left(\frac{n_k^w}{\sum_{j=1}^{K} n_j^w}\right), \tag{6}$$

where E_w is the entropy of word *w*, and *K* is the number of clusters. For each cluster *k*, the words in this cluster with the lowest entropy (i.e., the words that occur in the least number of clusters) are listed as distinctive words.

We have

$$E_k^w = E_w | n_k^w > 0, \tag{7}$$

where E_k^w is the entropy of word *w* in cluster *k*. We rank all words in cluster *k* according to E_k^w and present the highest-ranked words with the minimum E_k^w as the most distinctive words.

Together, F_k^w and E_k^w identify the most representative and distinctive words for each cluster, describing the contents of that cluster.

3.4 Ranking of articles within clusters

The ranking of articles is done using citation statistics in several different ways [6]. We use the statistics provided by Thomson Reuters' WOS. With these statistics, we can rank all articles based on the interest that other scientists have expressed according to their citations.

Our focus is on finding the most important articles within the clusters. This is done independently for each cluster by ranking all its articles. The ranking results from several independent methods with measures that perform alternative assessments. We start by calculating the number of citations for an article during each of the preceding six years. We then define four different impact measures based on citation impact and citation trends for all articles. Using the four measures, the articles within each cluster are assigned four alternative impact rankings that are then aggregated into a total ranking. The aggregation of the four rankings is designed to maximize robustness such that no single method dominates the final ranking. The process is repeated independently for each cluster. The four impact measures and the aggregated total rank are described in detail below.

3.4.1 Impact measures

The first measure is called *Impact1*. With this measure, we can rank all articles within a cluster according to the number of times they have been cited in the WOS database over the past year (i.e., the preceding 365 days). This can be done by the operator *citingArticles* in the WOS API.

We have

$$s_{1j}^{k} = citingArticles\big(A_j, timeSpan.begin, timeSpan.end\big).recordsFound, \quad (8)$$

where s_{1j}^{k} is the number of citations of A_j (i.e., the numerical value of the impact measure *Impact1*), A_j is the *j*th article in the search, *k* is the cluster position of A_j, *timeSpan.end* is today's date, *timeSpan.begin* is one year prior, and *recordsFound* calculates the number of found records. The highest-ranked article A_j is the one with the maximum value of s_{1j}^{k} for all $\{A_l\}$.

The second impact measure is called *Impact5*. This measure is similar to *Impact1*, except that it includes all citations over the past five years.

We have

$$s_{5j}^{k} = citingArticles\big(A_j, timeSpan.begin, timeSpan.end\big).recordsFound, \quad (9)$$

where s_{5j}^{k} is the number of citations of A_j over the past five years, and *timeSpan.begin* is five years prior to *timeSpan.end*. With *Impact5*, we rank all articles in the second ranking independently from the ranking made with *Impact1*.

The third impact measure is called *ImpactAIS*. Similarly, to *Impact5*, this measure uses citation statistics from the past five years. It is extended by weighting the source according to the source's importance with the Article Influence Score (AIS). However, AIS is only available for journals and unavailable for conferences, which therefore receive a zero weight. This may mean that with *ImpactAIS*, we will not find completely new articles that quickly receive many citations in conference proceedings. For articles that have been available for a few years, however, this should be a better way to rank. We need both methods because neither *Impact5* nor *ImpactAIS* is always the best method.

AIS is a measure developed to quantify the importance of a journal. Formally, it measures the average influence of the journal's articles during the first five years after publication. AIS is calculated by multiplying the journal's *EigenfactorScore*[3] (EFS) by 0.01 and dividing by the number of articles in the journal, normalized with respect to all articles in all publications covered by the Journal Citation Reports[4] (JCR). For example, since the average for all JCR journals is 1.0, an AIS score of 2.0 means that the average article in this journal has twice the influence of the average article in the JCR.

[3] http://www.eigenfactor.org/about.php (March 2019).
[4] http://www.webofknowledge.com/JCR (March 2019).

We have

$$AIS = 0.01 \times EFS/X, \tag{10}$$

where X is the number of articles in the journal published during five years divided by the number of articles in all JCR journals during five years.

EFS in (10) is based on the number of times articles in the journal, published in the past five years, have been cited in the JCR. It also takes into account the journal in which these citations occurred, so that citations in articles published in highly cited journals have a larger effect than do those in journals that are not as highly cited. In addition, references from an article in a journal to another article in the same journal are eliminated. For our purpose, we do not need to calculate EFS or AIS because Thomson Reuters provides AIS for all JCR journals.

We have

$$s_{\mathrm{AIS}j}^{k} = \sum_{A_l \in Y_j} AIS(A_l), \tag{11}$$

where $s_{\mathrm{AIS}j}^{k}$ is the number of citations in the preceding five years of A_j, where each citation is weighted by AIS of the citing source, and

$$Y_j = citingArticles\big(A_j, timeSpan.begin, timeSpan.end\big).records \tag{12}$$

is the set of citing articles in the past five years, where $timeSpan.begin$ is five years ago, and $timeSpan.end$ is today.

The fourth impact measure is called *ImpactReg*. This method performs a least-squares fit of a line to data on five-year citations changes (based on six years of data) for each article. The method ranks all articles according to the average change in citations during these five years, as defined by the slope of the regression line, as shown in Fig. 6.

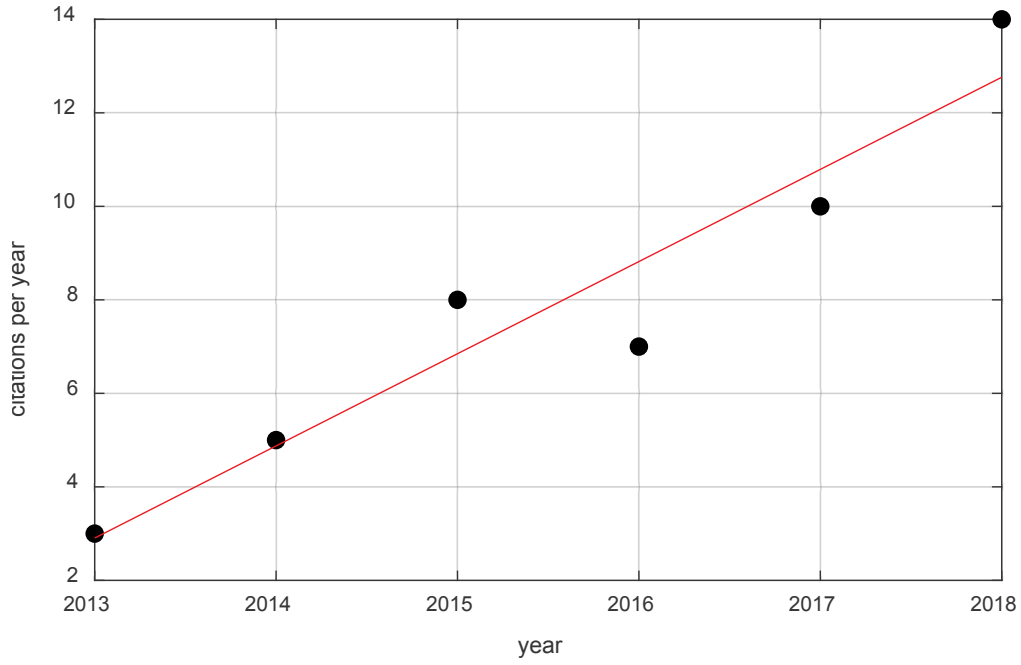


Figure 6. Example of a regression line (red line) over six years and the number of citations (black dots) per year for a particular article. The regression line in the example has a slope of 2.9047. This means that the number of citations for each year increases by an average of 2.9047 for this article. We use this value when we rank this article against all other articles in the same cluster.

The purpose of this method is to capture new articles with a strong trend that have not yet received enough citation coverage to receive high rankings by *Impact1*, *Impact5*, and *ImpactAIS*.

We have

$$
\begin{aligned}
s^{k}_{\mathrm{Reg}j} = Slope\{RegressionLine[& \\
& citingArticles\big(A_j, timeSpan_6.begin, timeSpan_6.end\big).recordsFound, \\
& citingArticles\big(A_j, timeSpan_5.begin, timeSpan_5.end\big).recordsFound, \\
& citingArticles\big(A_j, timeSpan_4.begin, timeSpan_4.end\big).recordsFound, \\
& citingArticles\big(A_j, timeSpan_3.begin, timeSpan_3.end\big).recordsFound, \\
& citingArticles\big(A_j, timeSpan_2.begin, timeSpan_2.end\big).recordsFound, \\
& citingArticles\big(A_j, timeSpan_1.begin, timeSpan_1.end\big).recordsFound]\}, \qquad (13)
\end{aligned}
$$

where $s^{k}_{\mathrm{Reg}j}$ is the slope of the regression line of six data points where $timeSpan_i.begin$ is the *i*th year before today, and $timeSpan_i.end$ is a year later. We use $s^{k}_{\mathrm{Reg}j}$ in cluster k as our fourth independent ranking of all articles A_j in the cluster.

In the next section, we will combine the four rankings derived in this section into a complete overall ranking of all articles A_j in each cluster k.

3.4.2 Combining all impact measures for an overall ranking

The measures derived in the previous section capture different aspects of scientific impact. The aggregated ranking should be able to reflect all these different aspects. A fairly good ranking by all four measures should result in a fairly good aggregated ranking. Furthermore, to receive an acceptable aggregated ranking index, it should be sufficient for an article to have an excellent ranking by one measure, even if the rankings by the other measures are mediocre. Finally, to ensure robust sampling, we want to eliminate any skewness in the

distribution for a particular measure. In what follows, we derive a method for aggregating the four impact measures into an overall ranking that meets these criteria.

When selecting *r* articles for further study from *m* articles ($r \leq m = |\{A_j\}|$) contained in cluster *k*, we use the four impact measures calculated for the articles in that cluster. For each impact measure, *Impact1*, *Impact5*, *ImpactAIS*, and *ImpactReg*, we sort all articles $\{A_j\}$ in cluster *k* in the decreasing order of impact according to $\{s_{ij}^k\}_j$ and renumber all articles within this cluster in the same decreasing order. Thus, the first article (A_1) has the highest impact according to s_{i1}^k within the current cluster *k*.

We assign a ranking score to *r* selected articles $\{A_j\}_{j=1}^r$. For article A_j in cluster *k* that received the *j*th highest $\{s_{ij}^k\}_j$, we calculate a ranking score with label P_{ij}^k determined according to

$$P_{ij}^k = \frac{r-j+1}{\sum_{l=1}^r l} = \frac{r-j+1}{\frac{1}{2}r(r+1)}, \tag{14}$$

where $i = \{1,5,AIS,Reg\}$, and *j* is the index of article A_j in position *j* in the ranking of all articles. If $r < j \leq m$, then $P_{ij}^k = 0$ applies by definition.

Since

$$\sum_{j=1}^{r} P_{ij}^k = 1 \tag{15}$$

we can consider $\{P_{ij}^k\}_j$ as a probability distribution where P_{ij}^k is the probability that A_j is the most preferred article according to impact measure *i*. This approach turns out to be immediately useful: for rankings that take into account more than one measure, we will use the calculated $\{P_{ij}^k\}_j$ instead of $\{s_{ij}^k\}_j$ because the former is more robust, as some bias in the distribution of $\{s_{ij}^k\}_j$ is eliminated, since $\{P_{ij}^k\}_j$ decreases linearly for all $\{s_{ij}^k\}_j$, as shown in Fig. 7.

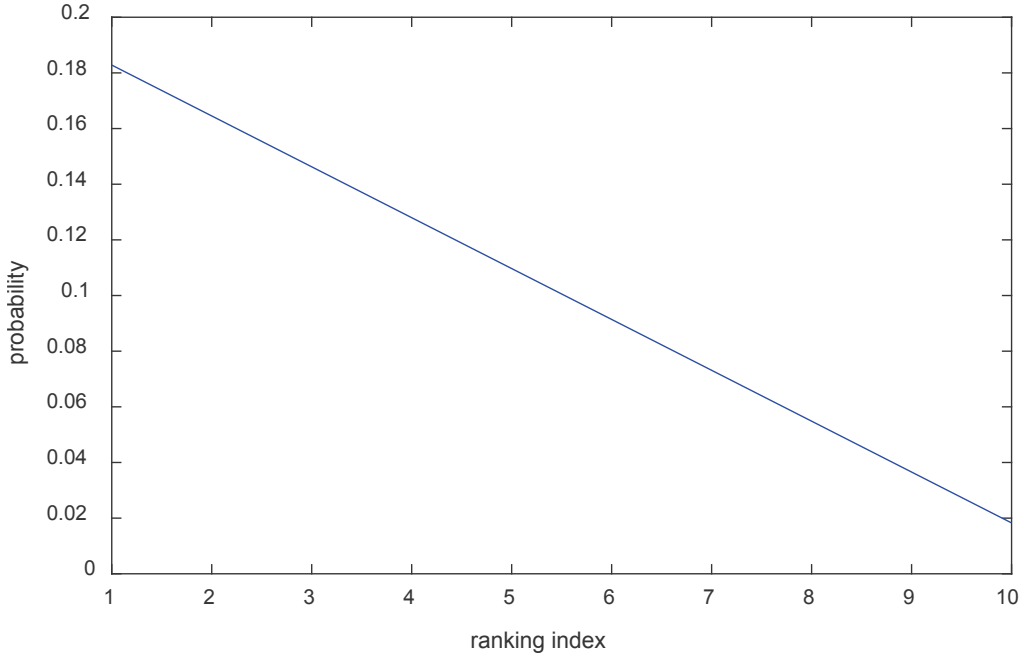


Figure 7. Example showing the probability $\{P_{ij}^k\}_j$ for 10 ranked articles with $1 \leq j \leq d = 10$. By switching from the initial measurements calculated by the methods of $\{s_{ij}^k\}_j$ with $i = \{1, 5, AIS, Reg\}$ to these ranking measures, we can calculate an overall ranking for each article based on all methods.

Consequently, we substitute in place of scores $\{s_{ij}^k\}_j$ for each measure the corresponding ranking scores $\{P_{ij}^k\}_j$ and calculate the probabilistic sum of all ranking scores $\{P_{ij}^k\}_i$ for each article A_j. This will be the total measure we use for the final ranking of articles in each cluster.

Within each cluster, we have so far had four different numberings with an individual numbering for each impact measure, since we have sorted all articles separately according to $\{s_{ij}^k\}_j$. We now number all articles within each cluster such that j always refers to the same article A_j for $\{P_{ij}^k\}_j$ and $\{s_{ij}^k\}_j$.

Finally, we calculate the total ranking score $\{P_{kj}^{Total}\}_j$ for each article A_j. We obtain

$$P_{kj}^{Total} = 1 - \prod_{i \in \{1,5,AIS,Reg\}} (1 - P_{ij}^k) \tag{16}$$

for each article A_j and cluster k, where P_{ij}^k is the ranking score of article A_j according to measure $i \in \{1, 5, AIS, Reg\}$. This is the probabilistic sum of all $\{P_{ij}^k\}_i$ [7]. Since each ranking score $\{P_{ij}^k\}_i \in [0, 1]$, the probabilistic sum thereof also belongs to interval [0, 1].

This is our final ranking of articles in cluster k. Ranking for every other cluster is done separately the same way. We can now select the highest-ranked articles within each cluster for further study, as shown in Fig. 1.

4 System description

In this section, we provide an overview of the horizon scanning software HSTOOL. HSTOOL is a web application built mainly in Scala[5]. The functionalities of the software correspond to the proposed workflow (Section 2) and include

1. Topic search in WOS,
2. Downloading of article records to a local database,
3. Clustering of articles according to topics,
4. Ranking of articles within each cluster based on scientometric impact, and
5. Outputting the resulting ranked clusters for further study.

4.1 Architecture

HSTOOL is built using the Play Framework[6] that follows the Model-View-Controller (MVC) architectural pattern shown in Fig. 8. Here, the Model is the representation of the information on which the application operates. This layer contains classes that describe article records and impact measures. The View layer provides a user interface that enables interaction with the Model, such as downloading search results and clustering and ranking of articles. The Controller layer processes user actions and updates the Model and View accordingly using a set of packages for communication via the WOS APIs as well as with the local database and for clustering and ranking calculations. The View layer constitutes the frontend of the application, while the Controller and Model layers constitute the backend.

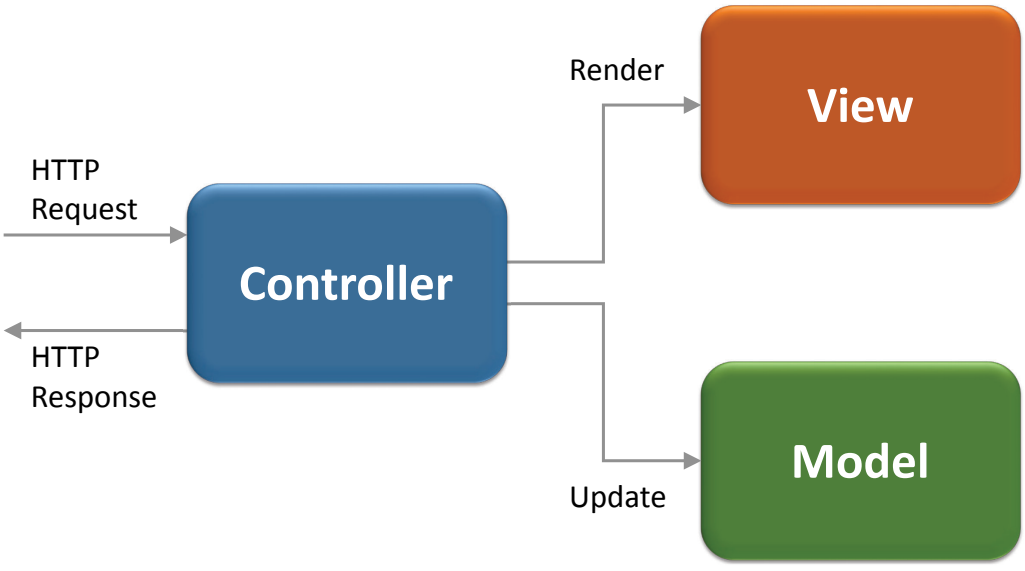


Figure 8. MVC architectural pattern.

[5] https://www.scala-lang.org (March 2019).
[6] https://www.playframework.com/ (March 2019).

4.2 System overview

Fig. 9 provides a system overview and illustrates the data flow in HSTOOL. HSTOOL communicates with the online scientific databases of WOS as well as a local PostgreSQL database where downloaded records and calculation results are stored.

The user interface of HSTOOL is shown in Fig. 10. In the following sections, the functionality listed above is described in further detail.

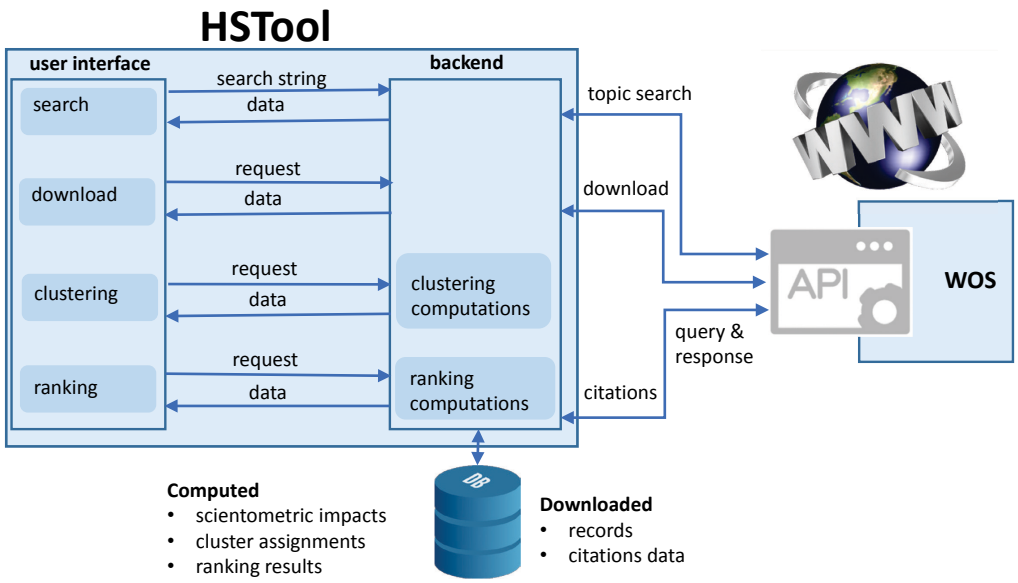


Figure 9. HSTOOL: System description and data flow.

Figure 10. HSTOOL user interface. Various functionalities are highlighted in red.

4.3 Connecting to Web of Science

HSTOOL connects to WOS through APIs provided by Thomson Reuters under a license agreement. The APIs allow web service operations[7], such as topic searching, and retrieval of article records, citation data and records of citing articles for a particular article.

4.4 Searching and downloading

Topic searches with HSTOOL are performed by combining search terms with logical operators. When a topic search is performed, the number of articles found is displayed in the HSTOOL user interface along with the search string used. A button is available for downloading the search result to a local Postgres database. Records of each article in the search result are saved, including keywords, abstract, and scientific field data that will later be used for clustering.

4.5 Clustering

Pressing the button "Discover clusters" initiates the article clustering algorithm (see Section 3.2). For each article in the downloaded corpus, a representative text is constructed by combining title, abstract, keywords, subjects, headings, and subheadings provided in the records from WOS. These are the texts that are fed to the GSDMM algorithm.

Once the clustering has been completed, a list of the discovered clusters is displayed on a separate web page, represented by representative and distinctive words for the cluster (see Section 3.4) and a list of the included articles. The discovered clusters are also displayed in a scrollable list in the HSTOOL main view, from which clusters can be selected for ranking.

4.6 Ranking

To rank articles within a cluster, it is necessary to select the cluster from the scrollable list and click "Compute ranking." Once the ranking has been completed, a ranked list of articles, including the calculated ranking index of each article, is displayed in the HSTOOL main view. Details of the ranking algorithm are outlined in Section 3.4.

4.7 Output

Once the ranking of a cluster has been completed, the results can be exported by clicking the button "Export results." This action produces a CSV file, including the WOS ID, article title, abstract, keywords, subjects, headings, journal title, ISSN, AIS factor, the estimated impact factors (see Section 3.4), and the resulting ranking index.

4.8 Performance and limitations

4.8.1 Clustering

The time complexity of GSDMM is $O(KDL)$ [1], where L is the average length of a text in the article set. Since one clustering must be performed for each value of β to find the optimal setting, the total time complexity of clustering will be influenced by the choice of granularity of β as well. However, since clusterings for different values of β can be performed in parallel, the algorithm's time consumption is in fact determined by the computational power

[7] http://help.incites.clarivate.com/wosWebServicesExpanded/WebServicesExpandedOverviewGroup/Introduction.html (March 2019).

of the system on which HSTOOL is run. As an example, using a laptop with 2.70 GHz CPU and 16 GB of RAM, a clustering of 1000 articles for a single value of β takes less than a minute.

4.8.2 Ranking

There are a number of limitations on the data transfer via the WOS APIs, which specify upper limits on processing speed and on the number of articles that can be analyzed by HSTOOL. The limitations critical to the performance of HSTOOL are listed below:

- A maximum of five parallel sessions of the same user are allowed.
- A maximum of two queries per second can be submitted from a session.
- A maximum of 100 records (i.e., XML-formatted information for an article) can be retrieved in one query.
- A maximum of 2500 queries per session are allowed.
- A maximum of 100 100 records can be retrieved from the same search.

For HSTOOL, the limitation on the number of queries per second is the most critical factor affecting performance. Since the ranking calculations require seven queries per article, it takes 3.5 seconds to retrieve the data needed for one article from WOS. In other words, ranking a category containing 1000 articles will take approximately an hour.

5 Case study of artificial intelligence in military applications

In this section, we report findings and results of a case study carried out to validate the proposed methodology and software tool. The topic of the case study was chosen within the authors' field of expertise to facilitate the evaluation of the results of the horizon scanning process.

5.1 Topic search

A search was performed using a combination of rather broad concepts, aiming to capture articles related to artificial intelligence (AI) in the context of defense applications. We compiled a topic search string of the form

[AI terms] AND [defense terms].[8]

The search resulted in 1358 hits in the WOS Core Collection, with publication years ranging from 1991 to 2019. Fig. 11 shows the number of articles per year for the search result. We will refer to the set of discovered articles as the AI corpus in the following sections.

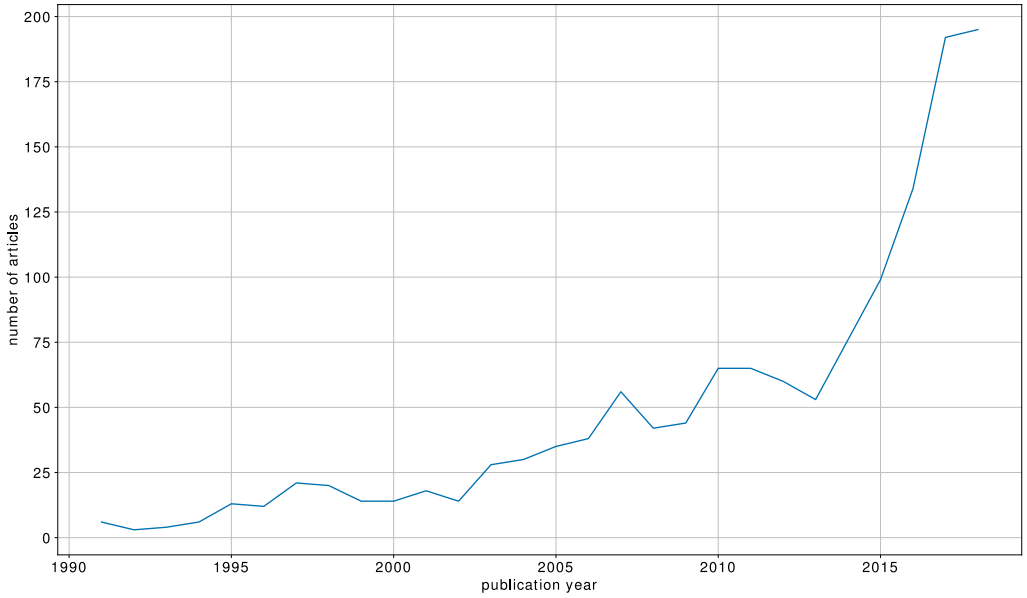


Figure 11. Number of articles per year in the AI corpus.

5.2 Clustering search results

Clustering of the search results encompasses two steps: first, determining the optimal value of parameter $\beta \in (0, 1)$ (which in turn yields the optimal number of clusters), and, second, performing the actual clustering with the optimal settings. The GSDMM algorithm clusters

[8] ("artificial intelligence" OR "machine learning" OR "deep learning" OR "neural network$") AND (military OR defense OR defence OR (command NEAR\1 control)) – the operator NEAR*n* signifies that the words on either side of it must be at most *n* words apart, and $ denotes the option of a plural s.

articles in the course of a set of Gibbs sampling iterations, during which the articles converge to a subset of the initial clusters. The size of this subset is determined by parameter settings. To understand how the algorithm works, we will first study the Gibbs sampling iterations for a fixed value of β, after which we will determine the value of β that yields the best clustering of the AI corpus.

5.2.1 Gibbs sampling iterations and convergence of entropy

At each Gibbs sampling iteration, the conditional probability p_{dki} given by equation (1) (see Section 3.2.1) is calculated for each article d and cluster k, yielding the probability that d is generated by k. Fig. 12 shows how p_{dki} varies over 15 Gibbs sampling iterations for a sample article. At first, the probability density function has spikes ($0 \ll p < 1$) at a few different clusters, but for most articles, it converges quickly towards a Dirac pulse ($p = 1$) at a certain cluster.

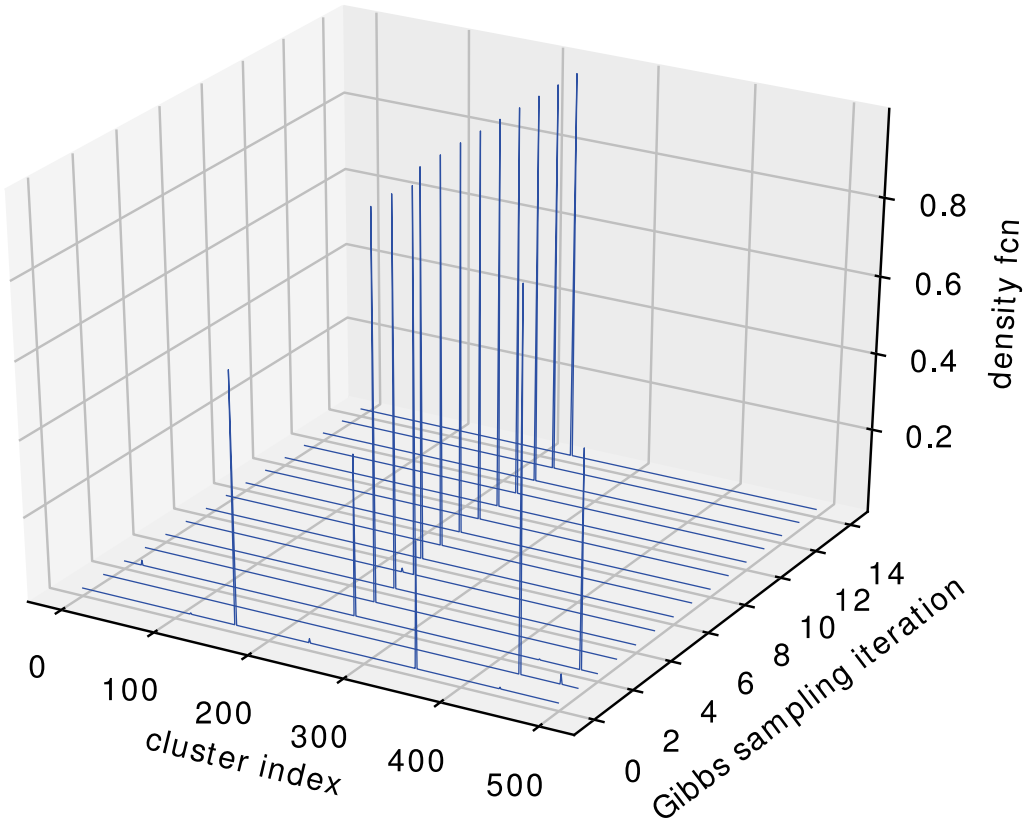


Figure 12. Probability density function for an article in the AI corpus, as it varies over the 15 Gibbs sampling iterations.

To study the convergence of the DMM algorithm, we calculate at each Gibbs sampling iteration i the entropy using (3). Fig. 13 shows how Ent_{di} for a set of articles that end up in the same cluster varies over 15 Gibbs iterations for a clustering run with $\beta = 0.101$. After 15 iterations, most articles have converged to low entropy, while a few may still oscillate between different clusters.

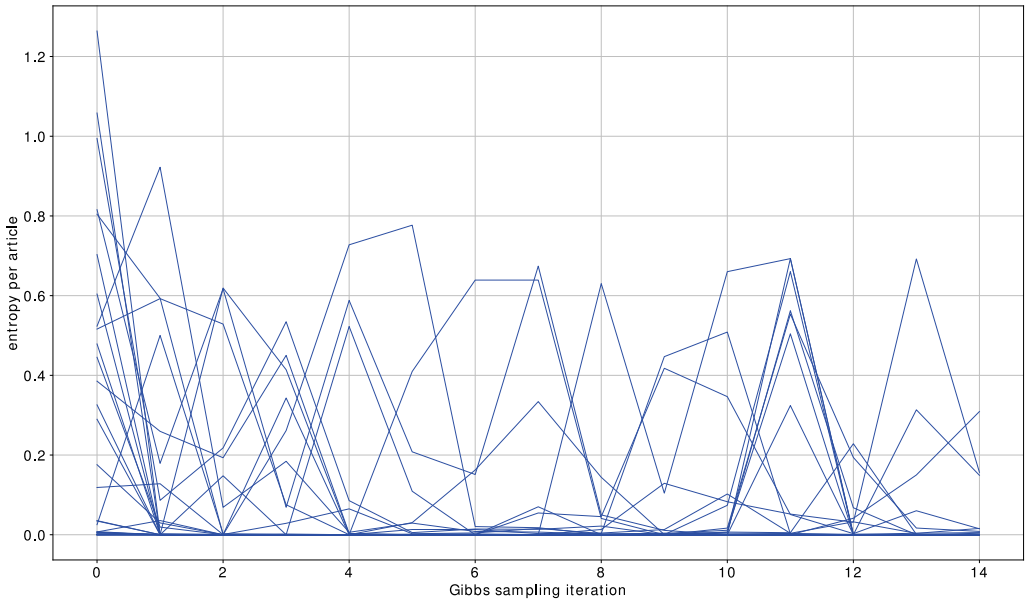


Figure 13. Example of how the entropies for a set of articles that end up in the same cluster vary over 15 Gibbs iterations.

Fig. 14 shows, for a subset of the entire corpus, how articles move between clusters during the Gibbs sampling iterations. Most articles converge quickly to their final clusters, while a few move around between clusters for a few iterations before settling. Some articles never quite converge and instead display an oscillating behavior.

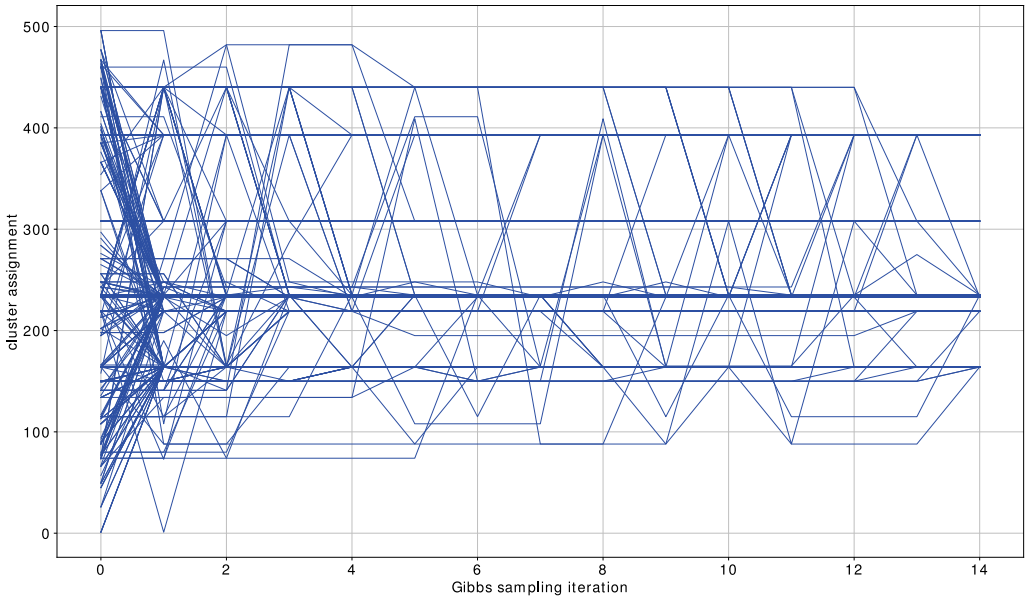


Figure 14. Paths of a set of articles from initialization to their final clusters over 15 Gibbs sampling iterations.

To determine the quality of a specific clustering (i.e., the clustering at a specific iteration i for some specified value of β), we calculate its entropy using (4) and the entire set of articles. Fig. 2 shows how the entropy for the entire AI corpus converges for a fixed value of β.

5.2.2 Determining the optimal number of clusters

Following the approach outlined in Section 3.2.2, the final entropy of the entire AI corpus for values of $\beta \in (0, 1)$ is calculated, and the value of β that minimizes the angle of the lower envelope curve is determined to be $\beta = 0.101$, yielding an average of 84 clusters over 100 runs, as shown in Fig. 3.

5.3 Analysis of clusters

We perform an individual clustering of the AI corpus using the optimal $\beta = 0.101$, this time yielding 90 clusters. The number of articles in the discovered clusters varies from 1 to 224. Fig. 15 shows the number of clusters of each size.

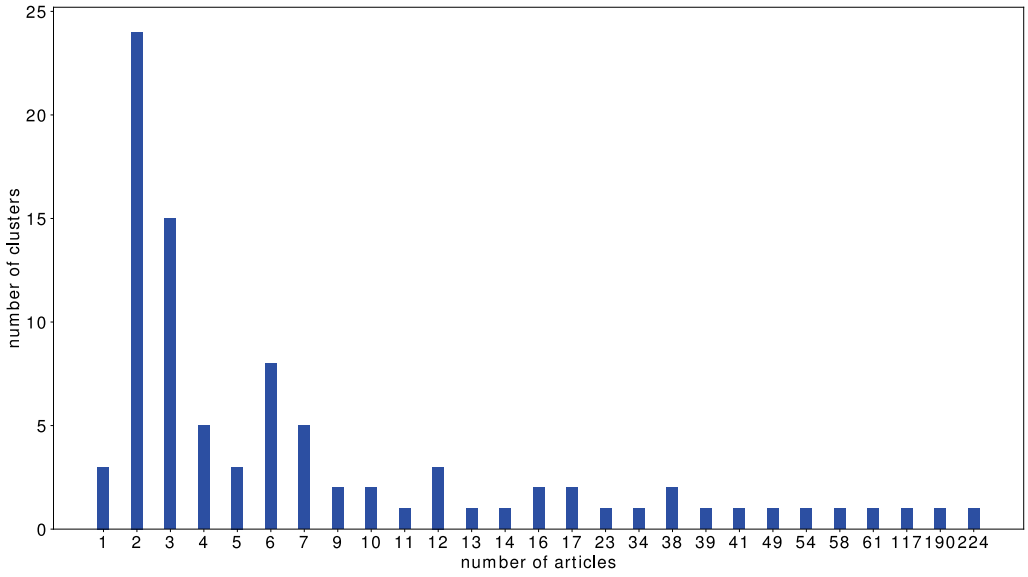


Figure 15. Number of clusters of each size for the AI corpus using DMM with $\beta = 0.101$.

To select clusters for further study, we use three criteria:

1. The cluster should be of sufficient size to represent a significant topic for the corpus.
2. The cluster should be well defined, i.e., have as low an entropy as possible.
3. The topic of the cluster should be relevant with respect to the original intention of the search query. Once a subset of clusters has been selected according to steps 1 and 2, clusters with irrelevant topics are removed from this subset.

Fig. 16 shows the number of articles in each of 90 discovered clusters. We will select clusters of size 15 or larger for further study. Fig. 17 shows the mean entropies for all such clusters. Among these, we examine the nine clusters with the lowest entropy in detail. The topics discovered using the method of Section 3.3 for these clusters are listed in Table 1.

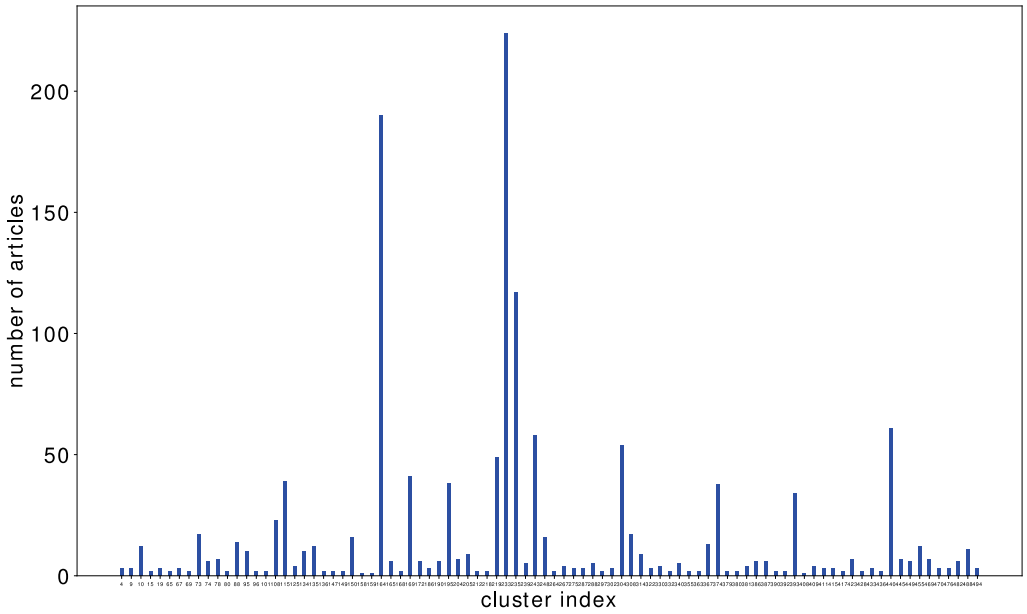


Figure 16. Number of articles in each of 90 discovered clusters.

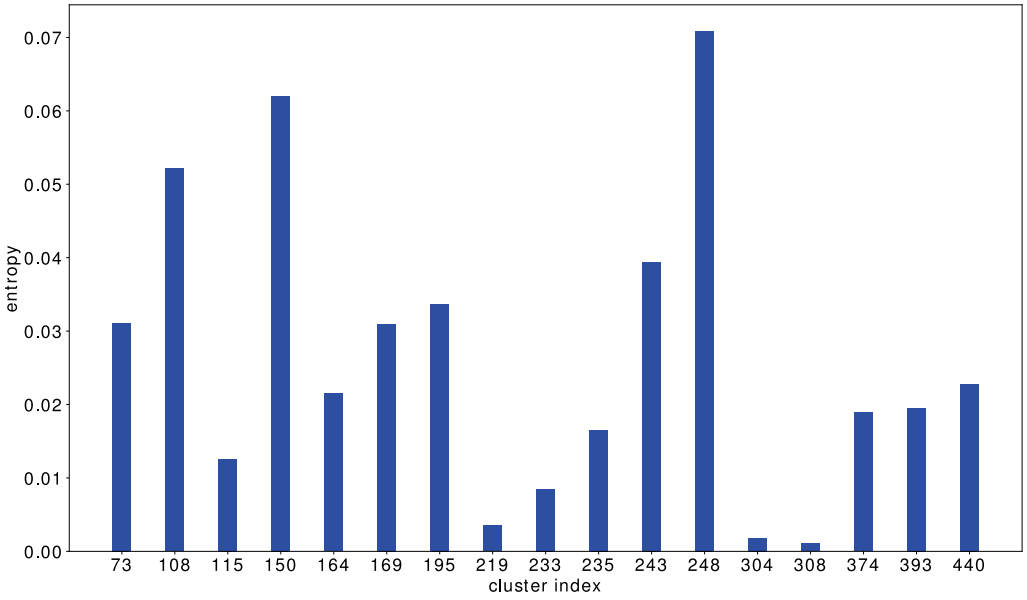


Figure 17. Entropies for all clusters of size 15 or larger.

The search queries used for topic search in WOS aimed to find articles on AI applications in the military domain. According to the discovered topics, clusters 115, 304 and 374 are beyond the scope of the study and are removed before further analysis. We are left with a set of six clusters that are sufficiently large, well defined and represent relevant topics. The descriptive and distinctive words of the six remaining clusters are shown in Table 1.

Table 1. Discovered topic words in the clusters chosen for further study. Clusters deemed irrelevant to the search query have been grayed out.

Id	Common words	Distinguishing words	Size
115	learning, system, education, computer, student	pupil, pill, installing, educate, math, leaming, pedagogical	39
164	image, target, network, recognition, neural	correlator, mstar, foreground, dividing, uncorrelated, eo, replaces	190
219	classification, signal, network, neural, feature	amc, instantaneous, cepstral, mel, mfcc, awgn, warped	49
233	attack, system, network, detection, computer	multicore, bodyguard, port/protocol, dsvms, dsvm, nash, protocol/internet	224
235	system, computer, agent, intelligence, decision	practically, illustration, bdi, nec, ner, automates, succession	117
304	cell, gene, immune, system, network, neural	hormone, overexpression, transcriptome, transgenic, glial, neurone	54
308	game, player, computer, artificial, defense	offense, beginner, dda, neuroevolution, shogi, warcraft	17
374	peptide, antimicrobial, machine, amp, learning, model	outlook, antitumor, staphylococcus, aureus, mic, insecticide	38
393	network, sensor, system, application, neural	ndia, biomechanical, fence, relay, transceivers, alcohol, steganography	34

Fig. 18 and Fig. 19 show, respectively, the number of articles in each of the studied clusters, and the number of citations of articles in each cluster over time. It can be noted that even though cluster 233 (attack, system, network) has the most articles and the strongest publications trend, the most cited cluster is cluster 164 (image, target, network), a trend that has been strong over the past 15 years and is still holding. It should be noted that the number of citing articles is based on all citing articles, whether part of the search result or not. A conclusion that can be drawn from Fig. 18 and Fig. 19 is that computer vision applications (cluster 164) remain a dominating topic within the "AI for the military" field, while defense against adversarial attacks for neural networks (cluster 233) has gained interest over the past few years. It should, however, be noted that this cluster, due to the design of our topic search, contains a certain number of articles about defense against such threats as malware attacks, which are not necessarily connected to military applications. We will therefore choose the computer vision cluster 164 as an example for further study.

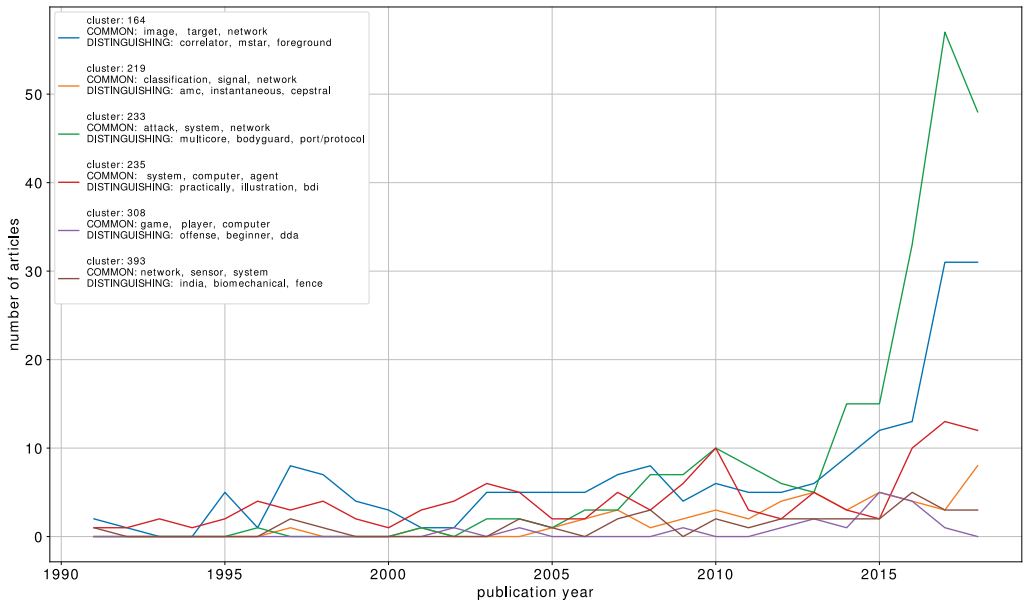


Figure 18. Number of articles per year in the clusters of interest.

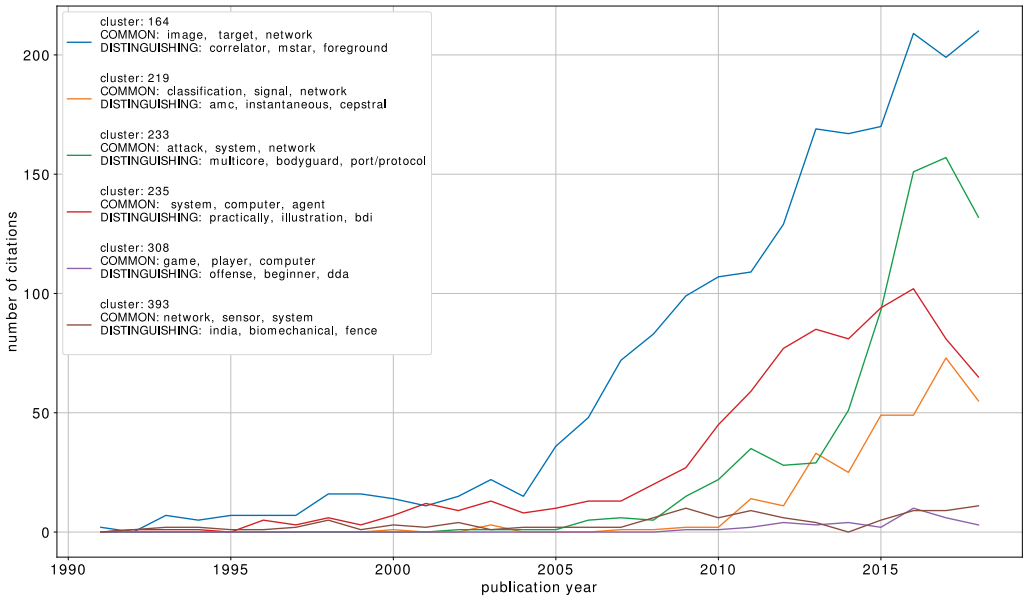


Figure 19. Number of citations per year generated by the clusters of interest.

All articles in the computer vision cluster are ranked according to the four impact measures in Section 3.4.1. The top results are shown in Table 2–5.

Table 2. Top 10 ranked articles in the computer vision cluster, considering all impact measures, see equation (16).

Ranking	Title	Year
1	Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition	2015
2	Remote Sensing Scene Classification by Unsupervised Representation Learning	2017
3	Learning Race from Face: A Survey	2014
4	Neural networks for automatic target recognition	1995
5	Adaptive fusion method of visible light and infrared images based on non-subsampled shearlet transform and fast non-negative matrix factorization	2014
6	A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs	2017
7	Modern Trends in Hyperspectral Image Analysis: A Review	2018
8	S-CNN-Based Ship Detection From High-Resolution Remote Sensing Images	2016
9	A neuromorphic system for video object recognition	2014
10	Airport Detection Based on a Multiscale Fusion Feature for Optical Remote Sensing Images	2017

Table 3. Top 10 ranked articles in the computer vision cluster, according to *Impact1*.

Imp 1	Title	Year	Overall ranking
95	Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information	2004	39
29	Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition	2015	1
26	Remote Sensing Scene Classification by Unsupervised Representation Learning	2017	2
11	Learning Race from Face: A Survey	2014	3
8	Modern Trends in Hyperspectral Image Analysis: A Review	2018	7
7	A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs	2017	6
4	Adaptive fusion method of visible light and infrared images based on non-subsampled shearlet transform and fast non-negative matrix factorization	2014	5
4	Neural networks for automatic target recognition	1995	4
3	S-CNN-Based Ship Detection From High-Resolution Remote Sensing Images	2016	8
3	Airport Detection Based on a Multiscale Fusion Feature for Optical Remote Sensing Images	2017	10

Table 4. Top 10 ranked articles in the computer vision cluster, according to *Impact5*.

Imp 5	Title	Year	Overall ranking
592	Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information	2004	39
81	Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition	2015	1
38	Remote Sensing Scene Classification by Unsupervised Representation Learning	2017	2
32	Learning Race from Face: A Survey	2014	3
28	Statistical pattern recognition in remote sensing	2008	50
19	Adaptive fusion method of visible light and infrared images based on non-subsampled shearlet transform and fast non-negative matrix factorization	2014	5
17	Neural networks for automatic target recognition	1995	4
10	S-CNN-Based Ship Detection From High-Resolution Remote Sensing Images	2016	8
10	Real-time automated counterfeit integrated circuit detection using x-ray microscopy	2015	14
9	A target-based color space for sea target detection	2012	57

Table 5. Top 10 ranked articles in the computer vision cluster, according to *ImpactAIS*.

Imp AIS	Title	Year	Overall ranking
83.93	3-D Object Recognition Using Bipartite Matching Embedded In Discrete Relaxation	1991	12
52.88	Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information	2004	39
48.56	Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition	2015	1
41.56	Remote Sensing Scene Classification by Unsupervised Representation Learning	2017	2
39.00	Statistical pattern recognition in remote sensing	2008	50
28.27	Neural networks for automatic target recognition	1995	4
17.25	Learning Race from Face: A Survey	2014	3
13.49	Detection of leukocytes in contact with the vessel wall from in vivo microscope recordings using a neural network	2000	32
11.56	Color machine vision for autonomous vehicles	1998	60
10.72	Texture synthesis and pattern recognition for partially ordered Markov models	1999	179

In the overall ranking (Table 6), a set of representation learning and object recognition articles receives the highest scores. This is a reasonable result, given the impressive progress made recently with regard to these topics. As discussed in Section 3.4.2, the different impact measures aim to capture different impact aspects. Examining the ranking results for the computer vision cluster, we note that *ImpactAIS* – that takes into account the AIS score of the journal an article is cited in – is the measure most in disagreement with the total ranking. As *ImpactAIS* does not in general reward recent articles, this would be an expected result, given the rapid development in the field of AI for computer vision (Fig. 18 and Fig. 19).

Table 6. Top 10 ranked articles in the computer vision cluster, according to *ImpactReg*. *Overall ranking* is listed in the right-most column.

Imp reg	Title	Year	Overall ranking
6.20	Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition	2015	1
4.74	Remote Sensing Scene Classification by Unsupervised Representation Learning	2017	2
1.94	Learning Race from Face: A Survey	2014	3
1.23	Adaptive fusion method of visible light and infrared images based on non-subsampled shearlet transform and fast non-negative matrix factorization	2014	5
1.14	Modern Trends in Hyperspectral Image Analysis: A Review	2018	7
1.09	A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs	2017	6
0.97	S-CNN-Based Ship Detection From High-Resolution Remote Sensing Images	2016	8
0.51	Neural networks for automatic target recognition	1995	4
0.51	Airport Detection Based on a Multiscale Fusion Feature for Optical Remote Sensing Images	2017	10
0.51	Kalman Filter Based Multiple Objects Detection-Tracking Algorithm Robust to Occlusion	2014	11

Notably, an article "*Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information"* that scores in the top two according to *Impact1*, *Impact5*, and *ImpactAIS* is not among the overall top ten ranked articles. This finding is observed because even though this article has been cited frequently, both during the past year and in the aggregate over five years, and cited in high AIS journals, its citation trend has been that of a steep decrease in the past years, yielding it the lowest rank of all articles for the *ImpactReg* measure.

The case study indicates that the proposed horizon scanning methodology and tool are useful for finding trending and significant topics in the scientific literature. The top-ranked articles within the studied cluster cover topics that have received significant attention in recent years, which validates the soundness of the impact measures and the aggregation method.

6 Conclusions

We have developed new methods for horizon scanning, integrated them with an existing clustering method, and implemented all methods in a system for horizon scanning of scientific literature to discover scientific trends. In particular, we have developed methods for finding an optimal number of clusters by developing an entropy-based method that focuses on the clustered articles rather than on the clusters themselves.

We conclude and show in a case study that with these methods, we can identify distinct clusters. These clusters can be categorized by automatically producing the most descriptive and distinctive words. Furthermore, we develop methods for a robust ranking of articles based on citation statistics and demonstrate in the case study how to produce an overall ranking of all articles in each category.

Overall, these methods automatically discover previously unknown categories, describe such categories with their most important words, rank all articles within each category by importance and deliver categories of ranked articles as the system output.

7 References

[1] Yin, J., Wang, J. (2014). A Dirichlet multinomial mixture model-based approach for short text clustering, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge discovery and data mining*. New York: ACM, pp. 233–242. doi:10.1145/2623330.2623715

[2] Shannon, C. E. (1948). A mathematical theory of communication, *The Bell System Technical Journal* **27**(3–4):379–423, 623–656. doi:10.1002/j.1538-7305.1948.tb01338.x

[3] Ahlberg, S., Hörling, P., Johansson, K., Jöred, K., Kjellström, H., Mårtenson, C., Neider, G., Schubert, J., Svenson, P., Svensson, P., Walter, J. (2007). An information fusion demonstrator for tactical intelligence processing in network-based defense, *Information Fusion* **8**(1):84−107. doi:10.1016/j.inffus.2005.11.002

[4] Nigam, K., McCallum, A. K., Thrun, S., Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM, *Machine Learning* **39**(2–3):103–134. doi:10.1023/A:1007692713085

[5] Rendon, E., Abundez, I., Arizmendi, A., Quiroz, E.M. (2011). Internal versus external cluster validation indexes, *International Journal of Computers and Communications* **5**(1):27–34. [Online] Available: http://www.naun.org/main/UPress/cc/20-463.pdf

[6] Mingers, J., Leydesdorff, L. (2015). A review of theory and practice in scientometrics, *European Journal of Operational Research* **246**(1):1–19. doi:10.1016/j.ejor.2015.04.002

[7] Schubert, J. (2012). Constructing and evaluating alternative frames of discernment, *International Journal of Approximate Reasoning* **53**(2):176–189. doi:10.1016/j.ijar.2011.09.009

FOI, Swedish Defence Research Agency, is a mainly assignment-funded agency under the Ministry of Defence. The core activities are research, method and technology development, as well as studies conducted in the interests of Swedish defence and the safety and security of society. The organisation employs approximately 1000 personnel of whom about 800 are scientists. This makes FOI Sweden's largest research institute. FOI gives its customers access to leading-edge expertise in a large number of fields such as security policy studies, defence and security related analyses, the assessment of various types of threat, systems for control and management of crises, protection against and management of hazardous substances, IT security and the potential offered by new sensors.

FOI
Defence Research Agency
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se