Journal of Artificial Intelligence Research x (xxxx) x-xx Submitted x/xx; published x/xx

Cross-lingual Document Similarity and Event Detection
TODO: Better title...

Jan Rupnik JAN.RUPNIK@IJS.SI
Andrej Muhič ANDREJ.MUHIC@IJS.SI
Gregor Leban GREGOR.LEBAN@IJS.SI
Primož Škraba PRIMOZ.SKRABA@IJS.SI
Blaž Fortuna BLAZ.FORTUNA@IJS.SI
Marko Grobelnik MARKO.GROBELNIK@IJS.SI
Artificial Intelligence Laboratory, Jožef Stefan Institute,
Jamova cesta 39, 1000 Ljubljana, Slovenia

Abstract

TODO: tralala hopsasa.

1. Introduction

Taken from XLite paper:Document retrieval is a well-established problem in data mining. There have been a large number of different approaches put forward in the literature. In this paper we concentrate on a specific setting: multi-lingual corpora. As the availability of multi-lingual documents has exploded in the last few years, there is a pressing need for automatic cross-lingual processing tools.

The prime example is Wikipedia - in 2001 the majority of pages were written in English, while in 2012, the percentage of English articles has dropped to 14%. In this context, we look at how to find similar documents across languages. In particular, we do not assume the availability of machine translators, but rather try to frame the problem such that we can use well-established machine learning tools designed for monolingual text-mining tasks. This work represents the continuation of previous work (?, ?, ?, ?) where we explored representations of documents which were valid over multiple languages. The representations could be interpreted as multi-lingual topics, which were then used as proxies to compute cross-lingual similarities between documents. We look at a specific aspect of this problem: the distribution of articles across languages in Wikipedia is not uniform. While the percentage of English articles has fallen, English is still the largest language and one of *hub* languages which not only have an order of magnitude more articles than other languages, but also many comparable articles with most of the other languages.

When doing document retrieval, we encounter two quite different situations. If for example, we are looking for a German article comparable to an English article, there is a large alignment between the document corpora (given by the intersection in articles in the two languages) making the problem well-posed. If, however, we look for a relevant Slovenian article given a Hindi article, the intersection is small, making the problem much harder. Since almost all languages have a large intersection in articles with *hub* languages, the question we ask in this paper is: can we exploit hub languages to perform better document retrieval between non-hub languages? A positive answer would vastly improve cross-lingual analysis

between less represented languages. In the following section, we introduce our representation followed by our experiments, which also shed light on the structural properties of the multilingual Wikipedia corpus.

2. Cross-lingual Document Similarity

Document similarity is an important component in techniques from text mining and natural language processing. Many techniques use the similarity as a black box, i.e., a kernel in Support Vector Machines. Comparison of documents (or other types of text snippets) is a well studied problem TODO: some references to prove that. In this section we define document similarity in a cross-lingual setting, where the similarity function receives documents in different languages. We conclude the section by an introduction of two datasets which we used in this paper to learn cross-lingual similarity functions.

2.1 Problem definition

Document representation.

Standard vector space model (?) represents documents as vectors, where each term corresponds to word or phrase in a fixed vocabulary. More formally, document d is represented by a vector $x \in \mathbb{R}^n$, where n corresponds to the size of the vocabulary, and vector elements x_k correspond to the number of times term k occurred in the document, also called *term frequency* or $TF_k(d)$.

We also used a term re-weighting scheme that adjusts for the fact that some words occur more frequently in general. A term weight should correspond to the importance of the term for the given corpus. The common weighting scheme is called *Term Frequency Inverse Document Frequency* ($TFIDF$) weighting. An *Inverse Document Frequency* (IDF) weight for the dictionary term k is defined as $\log(\frac{N}{DF_k})$, where DF_k is the number of documents in the corpus which contain term k. A document $TFIDF$ vector is its original vector multiplied element-wise by the weights.

The $TFIDF$ weighted vector space model document representation corresponds to a map $\phi : \text{text} \to \mathbb{R}^n$ defined by:

$$\phi(d)_k = TF_k(d) \log(\frac{N}{DF_k}).$$

Mono-lingual similarity. A common way of computing similarity between documents is *cosine similarity*,

$$sim(d_1, d_2) = \frac{\langle \phi(d_1), \phi(d_2) \rangle}{\|\phi(d_1)\| \|\phi(d_2)\|},$$

where $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ are standard inner product and Euclidean norm. Cosine similarity, and other related approaches, assumes that the similarity is reflected in the overlap of words, and as such works only when the documents d_1 and d_2 are written in the same language.

Cross-lingual similarity. Processing a multilingual dataset results in several vector spaces with varying dimensionality, one for each language. The dimensionality of the vector space corresponding to the i-th language is denoted by n_i and and the vector space model mapping is denoted by $\phi_i : \text{text} \to \mathbb{R}^{n_i}$. The similarity between documents in language i

and language j is defined as a bilinear operator represented as a matrix $S_{i,j} \in \mathbb{R}^{n_i \times n_j}$:

$$sim_{i,j}(d_1, d_2) = \frac{\langle \phi_i(d_1), S_{i,j}\phi_j(d_2) \rangle}{\|\phi_i(d_1)\| \|\phi_j(d_2)\|},$$

where d_1 and d_2 are documents written in the i-th and j-th language respectively. If the the maximal singular value of $S_{i,j}$ is bounded by 1, then the similarity scores will lie on the interval $[-1, 1]$. In section 3 we will describe some approaches to computing $S_{i,j}$ given training data.

2.2 Datasets

Taken from ITI paper:To investigate the empirical performance of the low rank approximations we will test the algorithms on a large-scale, real-world multilingual dataset that we extracted from Wikipedia by using inter-language links as an alignment. This results in a large number of weakly comparable documents in more than 200 languages. Wikipedia is a large source of multilingual data that is especially important for the languages for which no translation tools, multilingual dictionaries as Eurovoc, or strongly aligned multilingual corpora as Europarl are available. Documents in different languages are related with so called 'inter-language' links that can be found on the left of the Wikipedia page. The Wikipedia is constantly growing. There are currently four Wikipedias with more than 10^6 articles, 40 with more than 10^5 articles, 100 with more than 10^4 articles, and 216 with more than 1000 articles. Wikipedia uses special user-friendly markup language that is very easy to write but very hard to parse. Simplicity of language can cause ambiguities and moreover it is constantly changing. For example, separator — can be used in different contexts.

Wikipedia raw xml dumps of all currently 270 active editions were downloaded from the Wikipedia dump page. The xml files are too large to be parsed with DOM like parser that needs to store the whole xml tree in the memory, instead we implemented Sax like parser that tries to simulate behavior of Wikipedia official parser and is as simple, fast and error prone as possible. We parse all Wikipedia markup but do not extend the templates. Each Wikipedia page is embedded in the page tag. First we check if the title of the page consists of any special namespace and do not process such pages. Then we check if this is a redirection page and we store the redirect link as inter-language links can point to redirection link also. If nothing of the above applies we extract the text and parse the Wikipedia markup. Currently all the markup is removed.

We get inter-language link matrix using previously stored redirection links and inter-language links. If inter-language link points to the redirection we replace it with the redirection target link. It turns out that we obtain the matrix M that is not symmetric, consequently the underlying graph is not symmetric. That means that existence of the inter-language link in one way (i.e. English to German) does not guarantee that there is an inter-language link in the reverse direction (German to English). To correct this we transform this matrix to symmetric by computing $M + M^T$ and obtaining an undirected graph. In the rare case that we have multiple links pointing from the document, we pick the first one that we encountered. This matrix enables us to build an alignment across all Wikipedia languages.

3. Multilingual Latent Factor Models

3.1 Multilingual dataset

The cross-lingual similarity models presented in this paper are based on comparable corpora. That is, a corpus of documents in multiple languages, with alignment between documents that are of the same topic, or even a rough translation of each other. Wikipedia is an example of a comparable corpus, where a specific entry can be described in multiple languages (e.g. "Berlin" is currently described in 222 languages). News articles represent another example, where the same event can be described by newspapers in several languages.

More formally, *multilingual document* $d = (u_1, \dots u_m)$ is a tuple of m documents on the same topic (comparable), where u_i is the document written in language i. Note that individual document u_i can be an empty document (missing resource) and each d must contain at least two nonempty documents. A comparable corpus $D = d_1, \dots, d_N$ is a collection of multilingual documents. By using the vector space model we can represent D as a set of m matrices $X_1, \dots, X_m$, where $X_i \in \mathbb{R}^{n_i \times N}$ is the matrix corresponding to the language i and n_i is the vocabulary size of language i. Furthermore, let X_i^ℓ denote the ℓ-th column of matrix X_i and the matrices respect the document alignment - the vector X_i^ℓ corresponds to the TFIDF vector of the i-th component of multilingual document d_ℓ. We use N to denote the total row dimension of X, i.e. $N := \sum_{i=1}^{m} n_i$.

3.2 Algorithms

3.2.1 k-means

The k-means algorithm is perhaps the most well-known and used clustering algorithm. In order to apply the algorithm we first merge all the term-document matrices into a single matrix X by stacking the individual term-document matrices:

$$X := \left[X_1^T, X_2^T, \cdots, X_m^T\right]^T,$$

such that the columns respect the alignment of the documents (here MATLAB notation for concatenating matrices is used). Therefore, each document is represented by a long vector indexed by the terms in all languages.

We then run the k-means algorithm (?) and obtain a set of k centroid vectors, which form a matrix: $C := \left[C^1, \dots, C^k\right]$. The centroid matrix can be split vertically into m blocks:

$$C = [C_1^T \cdots C_m^T]^T,$$

according to the number of dimensions of each language, i.e. $C_i \in \mathbb{R}^{n_i}$.

Each matrix C_i represents a vector space basis and can be used to map points in $\mathbb{R}^{n_i}$ into a k-dimensional space, where the coordinates of a vector $x \in \mathbb{R}^{n_i}$ are expressed as:

$$(C_i^T C_i)^{-1} C_i^T x_i.$$

The resulting matrix (when appropriately scaled to have unit norm) for similarity computation between language i and language j is defined as:

$$C_i (C_i^T C_i)^{-1} (C_j^T C_j)^{-1} C_j.$$

In the implementation of k-means clustering we never form the matrix X explicitly but rather implement implicit multiplication as $Xx = \sum_{i=1}^{\ell} X_i x$ and multiplication with transpose as $X^T y = \sum_{i=1}^{\ell} X_i^T y_i$, where y is partitioned accordingly to X. That lowers memory requirement and compacts sparse indexing, leading to faster element access.

3.2.2 LSI

The next method is CL-LSI which is a variant of LSI (?) for more than one language. The method is based on computing a truncated singular value decomposition of $X \approx USV^T$. Since the matrix can be large we can use an iterative method like the Lanczos (?) algorithm with reorthogonalization to find the left singular vectors (columns of U) corresponding to the largest singular values. It turns out that the Lanczos method converges slowly as the gap between the leading singular values is small. Moreover, the Lanczos method is hard to parallelize. Instead we use a randomized version of the singular value decomposition (SVD) described in **??** that can be viewed as a block Lanczos method. That enables us to use parallelization and speeds up the computation considerably.

The cross-lingual similarity functions are based on a rank-k truncated SVD: $X \approx U\Sigma V^T$, where $U \in \mathbb{R}^{N \times k}$ are basis vectors of interest and $\Sigma \in \mathbb{R}^{k \times k}$ is truncated diagonal matrix of singular eigenvalues.

An aligned basis is obtained by first splitting U vertically according to the number of dimensions of each language: $U = [U_1^T \cdots U_m^T]^T$. Then, the same as with k-means clustering, we compute the pseudoinverses$P_i = (U_i^T U_i)^{-1} U_i^T$. The matrices P_i are used to change the basis from the standard basis in $\mathbb{R}^{n_i}$ to the basis of columns of U_i.

The numerical implementation of the least squares is done by QR algorithm, by computing factorizing $U_i = QR$, where $Q^T Q = I$ and R is triangular matrix. P_i is then obtained by solving $RP_i = Q$.

3.3 Hub languages

A *hub language* is a language with a high proportion of non-empty documents in $D = \{d_1, ..., d_\ell\}$, which in case of Wikipedia is English. We use the following notation to define subsets of the multilingual comparable corpus: let $a(i, j)$ denote the index set of all multilingual documents with non-missing data for the i-th and j-th language: $a(i, j) = \{k | d_k = (u_1, ..., u_m), u_i \neq \emptyset, u_j \neq \emptyset\}$, and let $a(i)$ denote the index set of all multilingual documents with non missing data for the i-th language.

Taken from XLite:

The first step in our method is to project $X_1, \ldots, X_m$ to lower dimensional spaces without destroying the cross-lingual structure. Treating the columns of X_i as observation vectors sampled from an underlying distribution $\mathcal{X}_i \in V_i = \mathbb{R}^{n_i}$, we can analyze the empirical cross-covariance matrices: $C_{i,j} = \frac{1}{|a(i,j)|-1} \sum_{\ell \in a(i,j)} (X_i^\ell - c_i) \cdot (X_j^\ell - c_j)^T$, where $c_i = \frac{1}{a_i} \sum_{\ell \in a(i)} X_i^\ell$. By finding low rank approximations of $C_{i,j}$ we can identify the subspaces of V_i and V_j that are relevant for extracting linear patterns between $\mathcal{X}_i$ and $\mathcal{X}_j$. Let X_1 represent the hub language corpus matrix. The standard approach to finding the subspaces is to perform the singular value decomposition on the full $N \times N$ covariance matrix composed of blocks $C_{i,j}$. If $|a(i, j)|$ is small for many language pairs (as it is in the case of Wikipedia), then many empirical estimates $C_{i,j}$ are unreliable, which can result in overfit-

ting. For this reason we perform the truncated singular value decomposition on the matrix $C = [C_{1,2} \cdots C_{1,m}] \approx USV^T$, where where $U \in \mathbb{R}^{n_1 \times k}, S \in \mathbb{R}^{k \times k}, V \in \mathbb{R}^{(\sum_{i=2}^{m} n_i) \times k}$. We split the matrix V vertically in blocks with $n_2, \ldots, n_M$ rows to obtain a set of matrices V_i: $V = [V_2^T \cdots V_m^T]^T$. Note that columns of U are orthogonal but columns in each V_i are not (columns of V are orthogonal). Let $V_1 := U$. We proceed by reducing the dimensionality of each X_i by setting: $Y_i = V_i^T \cdot X_i$, where $Y_i \in \mathbb{R}^{k \times N}$. The step is similar to Cross-lingual Latent Semantic Indexing (CL-LSI) (?)(?) which is less suitable due to a large amount of missing documents. Since singular value decomposition with missing values is a challenging problem and the alternative of replacing missing documents with zero vectors can result in degradation of performance.

The second step involves solving a generalized version of canonical correlation analysis on the matrices Y_i in order to find the mappings P_i. The approach is based on the sum of squares of correlations formulation by Kettenring (?), where we consider only correlations between pairs $(Y_1, Y_i), i > 1$ due to the hub language problem characteristic. Let $D_{i,i} \in \mathbb{R}^{k \times k}$ denote the empirical covariance and $D_{i,j}$ denote the empirical cross-covariance computed based on Y_i and Y_j. We solve the following optimization problem:

$$\underset{w_i \in \mathbb{R}^k}{\text{maximize}} \quad \sum_{i=2}^{m} \left(w_1^T D_{1,i} w_i\right)^2 \quad \text{subject to} \quad w_i^T D_{i,i} w_i = 1, \quad \forall i = 1, \ldots, m.$$

By using Lagrangian multiplier techniques (we omit the derivation due to space constraints TODO: Add what was omitted), the problem can be reformulated as the eigenvalue problem: $\left(\sum_{i=2}^{m} G_i G_i^T\right) \cdot V = \Lambda \cdot V$, where $G_i = H_1^T D_{1,i} H_i$ and $H_i = Chol(D_{i,i})^{-1}$ where $Chol(\cdot)$ is the Cholesky decomposition: $X = Chol(X)^T Chol(X)$. The vectors w_i can be reconstructed from the dominant eigenvector v, as: $w_1 = H_1 v$ and $w_i \propto H_i G_i^T v$ for $i > 1$ (w_i need to be appropriately normalized). Higher dimensional mappings W_i are obtained in a similar way by setting the constraint $W_i^T D_{i,i} W_i = I$, where I is the identity matrix. The constraint forces the columns of W_i to be uncorrelated (orthogonal with respect to covariance matrix). W_i can be extracted from a set of dominant eigenvectors of matrix V according to eigenvalues Λ. The technique is related to a generalization of canonical correlation analysis (GCCA) by Carroll(?), where an unknown group configuration variable is defined and objective is to maximize the sum of squared correlation between the group variable and the others. The problem can be reformulated as an eigenvalue problem. The difference lies in the fact that we set the unknown group configuration variable as the hub language, which simplifies the solution. The complexity of our method is $O(k^3)$, whereas solving the GCCA method scales as $O(N^3)$ where N is the number of samples (see (?)). Another issue with GCCA is that it cannot be directly applied to the case of missing documents.

To summarize, we first reduced the dimensionality of our data to k-dimensional features and then found a new representation (via linear transformation) that maximizes directions of linear dependence between the languages. The final projections that enable mappings to a common space are defined as: $P_i(x) = W_i^T V_i^T x$.

3.4 Scaling

100 languages!

3.5 Evaluation

3.5.1 Link prediction

3.5.2 Hub language

Taken from XLite paper:To investigate the empirical performance of our algorithm we will test it on a large-scale, real-world multilingual dataset that we extracted from Wikipedia by using so called 'inter-language' links as an alignment. We select a subset of Wikipedia languages containing three major languages, English–*en* (hub language), Spanish–*es*, Russian–*ru*, and five minority (in the sense of Wikipedia sizes) languages, Slovenian–*sl*, Piedmontese–*pms*, Waray-Waray–*war* (all with about 2 million native speakers), Creole–*ht* (8 million native speakers), and Hindi–*hi* (180 million native speakers). For preprocessing we remove the documents that contain less than 20 different words (stubs) and remove words occur in less than 50 documents as well as the top 100 most frequent words (in each language separately). We represent the documents as normalized TFIDF(?) weighted vectors. These are languages with Wikipedia article sizes comparable to the minority languages. (since the number of speakers does not directly correlate with the number of Wikipedia articles). The prime hub candidate is the English language which is well aligned with all other Wikipedia languages, although the alignment quality varies quite a bit. Furthermore, we remove the documents that contain less than 20 different words and remove words that are too infrequent as well as the top 100 most frequent words in the vocabularies. Furthermore, we call the document consisting of less than 20 different words, a stub. This documents are typically garbage, the titles of the columns in the table, remains of the parsing process, or Wikipedia articles with very little or no information contained in one or two sentences.

The evaluation is based on splitting the data into training and test sets (described later). On the training set, we perform the two step procedure to obtain the common document representation as a set of mappings P_i. A test set for each language pair, $test_{i,j} = \{(x_\ell, y_\ell)|\ell = 1 : n(i,j)\}$, consists of comparable document pairs (linked Wikipedia pages), where $n(i,j)$ is the test set size. We evaluate the representation by measuring mate retrieval quality on the test sets: for each ℓ, we rank the projected documents $P_j(y_1), \ldots, P_j(y_{n(i,j)})$ according to their similarity with $P_i(x_\ell)$ and compute the rank of the mate document $r(\ell) = rank(P_j(y_\ell))$. The final retrieval score (between -100 and 100) is computed as: $\frac{100}{n(i,j)} \cdot \sum_{\ell=1}^{n(i,j)} \left(\frac{n(i,j)-r(\ell)}{n(i,j)-1} - 0.5 \right)$. A score that is less than 0 means that the method performs worse than random retrieval and a score of 100 indicates perfect mate retrieval. The mate retrieval results are included in Table 1.

We observe that the method performs well on all pairs between languages: *en*, *es*, *ru*, *sl*, where at least 50,000 training documents are available. We notice that taking $k = 500$ or $k = 1000$ multilingual topics usually results in similar performance, with some notable exceptions: in the case of (*ht*, *war*) the additional topics result in an increase in performance, as opposed to (*ht*,*pms*) where performance drops, which suggests overfitting. The languages where the method performs poorly are *ht* and *war*, which can be explained by the quality of data (see Table 3 and explanation that follows). In case of *pms*, we demonstrate that solid performance can be achieved for language pairs (*pms*, *sl*) and (*pms*, *hi*), where only 2000 training documents are shared between *pms* and *sl* and no training documents are available between *pms* and *hi*. Also observe that in the case of (*pms*, *ht*) the method still

obtains a score of 62, even though training set intersection is zero and *ht* data is corrupted, which we will show in the next paragraph.

Table 1: Pairwise retrieval, 500 topics;1000 topics

	en	es	ru	sl	hi	war	ht	pms
en		98 - 98	95 - 97	97 - 98	82 - 84	76 - 74	53 - 55	96 - 97
es	97 - 98		94 - 96	97 - 98	85 - 84	76 - 77	56 - 57	96 - 96
ru	96 - 97	94 - 95		97 - 97	81 - 82	73 - 74	55 - 56	96 - 96
sl	96 - 97	95 - 95	95 - 95		91 - 91	68 - 68	59 - 69	93 - 93
hi	81 - 82	82 - 81	80 - 80	91 - 91		68 - 67	50 - 55	87 - 86
war	68 - 63	71 - 68	72 - 71	68 - 68	66 - 62		28 - 48	24 - 21
ht	52 - 58	63 - 66	66 - 62	61 - 71	44 - 55	16 - 50		62 - 49
pms	95 - 96	96 - 96	94 - 94	93 - 93	85 - 85	23 - 26	66 - 54	

We now describe the selection of train and test sets. We select the test set documents as all multi-lingual documents with at least one nonempty alignment from the list: (*hi*, *ht*), (*hi*, *pms*), (*war*, *ht*), (*war*, *pms*). This guarantees that we cover all the languages. Moreover this test set is suitable for testing the retrieval thorough the hub as the chosen pairs have empty alignments. The remaining documents are used for training. In Table 2, we display the corresponding sizes of training and test documents for each language pair. The first row represents the size of the training sets used to construct the mappings in low dimensional language independent space using the English–*en* as a hub. The diagonal elements represent number of the unique training documents and test documents in each language.

Table 2: Pairwise training:test sizes (in thousands)

	en	es	ru	sl	hi	war	ht	pms
en	671 - 4.64	463 - 4.29	369 - 3.19	50.3 - 2	14.4 - 2.76	8.58 - 2.41	17 - 2.32	16.6 - 2.67
es		463 - 4.29	187 - 2.94	28.2 - 1.96	8.72 - 2.48	6.88 - 2.4	13.2 - 2	13.8 - 2.58
ru			369 - 3.19	29.6 - 1.92	9.16 - 2.68	2.92 - 1.1	3.23 - 2.2	10.2 - 1.29
sl				50.3 - 2	3.83 - 1.65	1.23 - 0.986	0.949 - 1.23	1.85 - 0.988
hi					14.4 - 2.76	0.579 - 0.76	0.0 - 2.08	0.0 - 0.796
war						8.58 - 2.41	0.043 - 0.534	0.0 - 1.97
ht							17 - 2.32	0.0 - 0.355
pms								16.6 - 2.67

We further inspect the properties of the training sets by roughly estimating the fraction `rank(A)/min(size(A))` for each training English matrix and its corresponding mate matrix. Ideally, these two fractions are approximately the same so both aligned spaces should have reasonably similar dimensionality. We display these numbers as pairs in Table 3.

Table 3: Dimensionality drift

(en, de)	(en, ru)	(en, sl)	(en, hi)	(en, war)	(en, ht)	(en, pms)
(0.81, 0.89)	(0.8, 0.89)	(0.98, 0.96)	(1, 1)	(0.74, 0.56)	(1, 0.22)	(0.89, 0.38)

It is clear that in the case of Creole language only at most 22% documents are unique and suitable for the training. Though we removed the stub documents, many of remaining documents are almost the same, as the quality of some minor Wikipedias is low. This was confirmed for Creole, Waray-Waray, and Piedmontese language by manual inspection. The low quality documents correspond to templates about the year, person, town, etc. and contain very few unique words.

We also have a problem with the quality of the test data. For example, if we look at test pair (*war*, *ht*) only 386/534 Waray-Waray test documents are unique but on other side almost all Creole test documents (523/534) are unique. This indicates a poor alignment which leads to poor performance.

4. Cross-lingual Event Linking

One of the use cases where article linking information can be applied is also in an online system for detection of world events, called Event Registry(). Event Registry is a repository of events, where events are automatically identified by analyzing news articles. The term event is vague and ambiguous, but for the practical purposes we define it as "any significant happening that is being reported about in the media". Examples of events would include shooting down of the Malaysia Airlines plane over Ukraine on July 18th, 2014 and HSBC's admittance of aiding their clients in tax evasion on February 9th, 2015. Events such as these are covered by many articles and the question is how to find all the articles in different languages that are describing a single event.

Event Registry works by collecting and analyzing news articles that are published by news outlets in different languages all over the world. Collected articles are first semantically annotated by identifying in them mentions of relevant concepts – either entities or important keywords. The disambiguated and entity linking of the concepts is done using Wikipedia as the main knowledge base. The key step in Event Registry is then to apply an online clustering algorithm on the articles in order to identify groups of articles that are discussing the same event. For each new article, the clustering algorithm determines if the article should be assigned to some existing cluster or into a new cluster. The underlying assumption is that articles that are describing the same event are similar enough and will therefore be put into the same cluster.

In short, the article clustering works as follows (details are available in ()). First we create a new document by joining the title and the body of the article. We then tokenize the document, remove the stop words and stem the words. The remaining words are represented as a vector in the vector-space model. Each value in the vector is additionally normalized using TF-IDF. After normalization, we use the vector to find the most similar existing cluster. To be able to compare the vector with a cluster we first need a vector representation for each of the clusters. This is done by computing a centroid vector of the cluster by simply averaging the vectors of the articles in the cluster. By computing the cosine similarity of the article vector with each of the centroid vectors we then find the cluster with the highest similarity. If the highest similarity is above the selected threshold (we used 0.4 in our experiments), the article is assigned to the corresponding cluster, otherwise a new cluster is created, initially containing only the single article.

Since articles about an event are commonly written only for a short period of time, we remove clusters once the oldest article in the cluster becomes more than 4 days old. This housekeeping mechanism prevents the clustering from becoming slow and also ensures that articles are not assigned to obsolete clusters.

In order to treat the cluster as describing an event in Event Registry, the cluster has to grow above a certain number of articles (the number depends on the language). This condition ensures that only content that receives a sufficient coverage in the media is treated as an event. Once a new event is identified, a new unique ID is assigned to it and the main information about the event is then automatically extracted by analyzing the articles assigned to it. The extracted information includes properties such as the date of the event, the location, who is involved in it, what the event is about, etc.

The clustering of articles is done for each language separately – English articles are, for example, clustered separately from Spanish articles. When events are identified, they will therefore be identified from clusters in each language separately. As a result, the same event will be represented in Event Registry with two or more event IDs, and each event will only contain coverage in a single language. To avoid this problem we want to exploit the available information about the similarity of the articles in a cluster with articles in other languages. Using this information we hope to be able to connect clusters in different languages and represent them using a single event ID.

4.1 Problem definition

Formally, the problem of identifying the clusters describing the same event can be formalized as follows. Each article a_i is written in a language l, where $l \in L = \{l_1, l_2, ..., l_k\}$. For each language l, a set of clusters C_l is generated and maintained that only contains articles in language l. For any cluster $c_i \in C_{l_a}$, the task is then to identify other clusters $c_j \in C_{l_b}$ (where $l_a \neq l_b$) that are describing the same event as c_i. We will refer to cluster pairs c_i and c_j that are describing the same event as equivalent clusters. An additional requirement of the task is that identification of the equivalent clusters should not be done exhaustively – for a cluster c_i we wish to avoid testing all other clusters c_j. Performing exhaustive comparison for each cluster c_i would result in $O(N^2)$ tests, where N is the number of all clusters, which is not feasible when the number of clusters is in tens of thousands.

4.2 Algorithm

In order to identify clusters c_j that are equivalent to cluster c_i we have developed a procedure that consists of two main tasks. The first task is to efficiently identify a limited set of clusters that are potential candidates for being equivalent to c_i. The second task is then about performing a detailed analysis of the candidates and finding the ones that are equivalent to c_i.

The details of the first task are described in Algorithm 1. The algorithms starts by individually inspecting each article a_i in the cluster c_i. Using the CCA it then identifies 10 most similar articles to a_i in each language $l \in L$. For each identified similar article a_j we then identify cluster c_j to which the article belongs to and add the cluster to the set of candidates that should be more closely inspected. The assumption made by this algorithm is that if two articles in different languages are describing the same event, the

```
input  : test cluster c_i,
         a set of clusters C_l for each language l ∈ L
output: a set of clusters C that are potentially equivalent to c_i
C = {};
for article a_i ∈ c_i do
    for language l ∈ L do
        /* use CCA to find 10 most similar articles to article a_i in
           language l                                                  */
        SimArt = getCCASimArts(a_i, l);
        for article a_j ∈ SimArt do
            /* find cluster c_j to which article a_j is assigned to   */
            c_j := c; c ∈ C_l, a_j ∈ c;
            /* add cluster c_j to the set of candidates C             */
            C ← c_j;
        end
    end
end
```

Algorithm 1: Algorithm for identifying candidate clusters C that are potentially equivalent to c_i

CCA similarity between the documents will be higher than when comparing two articles about different events. Since the CCA similarity is only an estimate, the assumption does not always hold. Due to data redundancy, however, we do not consider this to be an issue. Since we are testing each article in the cluster and compute 10 most similar articles for each article, we are very likely to identify among the candidates also the clusters that are equivalent to cluster c_i.

The second part of the procedure is described in the Algorithm 2. The goal of this task is to determine which (if any) of the candidate clusters are equivalent to c_i. To solve the task we represent it as a supervised learning problem. For each candidate cluster $c_j \in C'$ we compute a vector of learning features that should be indicative of whether the c_i and c_j are equivalent or not. The details about these features will be described in the next section. Once the features are computed, we classify the example with a classification model that was trained using a training dataset of hand-labeled examples. If the model predicts that tested clusters are equivalent, the c_j is added to the set C'.

The learning dataset used to train a classification model consists of cluster pairs c_i and c_j for which a human annotator provided feedback whether the clusters are equivalent or not. The dataset contains XYZ examples, of which XYZ are equivalent clusters pairs and XYZ are not. To train a model, the vector of features was first extracted for each learning example in the same way as described in Algorithm 2. A linear SVM method was then used to train the classifier M.

input : test cluster c_i,
a set of candidate clusters C that are potentially equivalent to c_i
classification model M trained on the hand-labeled examples
output: a set of clusters C' that are equivalent to c_i
$C' = \{\}$;
for *cluster* $c_j \in C$ **do**
  `/* extract a vector of features for cluster pair` c_i `and` c_j `*/`
  $\vec{v} = f(c_i, c_j)$;
  `/* classify the feature vector` $\vec{v}$ `using the model` M `*/`
  $pred = M(\vec{v})$;
  if $pred = true$ **then**
    $C' \leftarrow c_j$
  end
end

Algorithm 2: Algorithm for identifying clusters C' that are equivalent to cluster c_i

4.2.1 Learning features

In order to use a supervised approach to determine if two clusters c_i and c_j are equivalent we need to extract from the cluster pair as many informative features as possible. Based on the availability of the data, we have extracted the following groups of features:

- **CCA related features**. CCA was already used to obtain candidates for the equivalent clusters. Additionally, it can also be used to provide valuable learning features. There are two main features that we extract using CCA – `linkCount` and `avgSimScore`. `linkCount` is the number of times an article $a_i \in c_i$ has as one of the 10 most similar articles an article $a_j \in c_j$. In other words, it is the number of times an article from c_i has a very similar article, which is in c_j. Beside the number of links between the two clusters we can also take into account the similarity scores of the links. The `avgSimScore` is the feature that represents the average similarity score of the links between the two clusters.

- **Concept related features**. As mentioned before, articles that are imported into Event Registry are first semantically annotated by linking mentioned entities and keywords to the corresponding Wikipedia pages. Whenever Barack Obama is, for example, mentioned in the article, the article is annotated with a link to his Wikipedia page (*http://en.wikipedia.org/wiki/Barack_Obama*). In the same way all mentions of people, locations, organizations and even ordinary keywords (e.g. bank, tax, ebola, plane, company) are annotated. Although the Spanish article about Obama will be annotated with his Spanish version of the Wikipedia page, we can almost in all cases normalize the Wikipedia links to their English versions. This can be done since Wikipedia itself provides information regarding which pages in different languages represent the same concept/entity. Using this approach, the word "avión" in a Spanish article will be annotated with the same concept as the word "plane" in an English article. Although articles are in different languages, the annotations can therefore provide a shared vocabulary that can be used to compare articles/clusters. By ana-

lyzing all articles in clusters c_i and c_j we can identify the most relevant entities and keywords for each cluster. Additionally we can also assign weights to the concepts based on how frequently they occur in the articles in the cluster. From the list of relevant concepts and corresponding weights we can now define a number of features. `entityCosSim` represents the cosine similarity on the vector of entities mentioned in both clusters. Similarly, `keywordCosSim` represents the cosine similarity on the keywords only. In case the weights are not relevant, we have also added two features (`entityJaccardSim` and `keywordJaccardSim`), where Jaccard similarity is used instead of the cosine similarity.

- **Miscellaneous features**. This group contains three miscellaneous features that seem discriminative but are unrelated to the previous two groups. The first feature is related to the event location. By analyzing the locations mentioned in the articles in a cluster we are able to estimate where the event occurred. The `hasSameLocation` feature is a boolean variable that is true when the location of the event in both clusters is the same. Another important feature is related to the time when the articles in the clusters were published. For articles in each cluster we first compute the average date and time when the articles were published. The `timeDiff` is then defined as the absolute difference in hours between the two computed dates. The last feature in this group is computed by analyzing the mentions of dates in the articles. Using an extensive set of regular expressions we are able to detect mentions of dates in different forms. To compute the feature `sharedDates` we start by identifying the list of dates mentioned in articles in each cluster separately. The value of `sharedDates` is then determined as the ratio of all dates that are detected in both clusters.

4.3 Evaluation

In order to evaluate the quality of individual features and the overall model we have performed a set of experiments.

evaluate accuracy on different language pairs only. compute accuracies using the CCA features only, using concept features only, or using all features.

4.3.1 Compare different feature subsets

4.3.2 Accuracy across different language pairs

5. Related work

5.1 Cross-lingual Document Similarity

5.2 Multilingual Latent Factor Models

Ask Wray

5.3 Cross-lingual Event Linking

6. Discussion

Acknowledgments

The authors wish to thank blah blah blah.