

Related Articles Discovery in Large Corpora

Master-Thesis von Ziyang Li aus Tianjin, China

Tag der Einreichung:

1. Gutachten: Prof. Dr. Iryna Gurevych
2. Gutachten: Nils Reimers



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Ubiquitous Knowledge Processing

Related Articles Discovery in Large Corpora

Vorgelegte Master-Thesis von Ziyang Li aus Tianjin, China

1. Gutachten: Prof. Dr. Iryna Gurevych
2. Gutachten: Nils Reimers

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den July 18, 2016

(Ziyang Li)

Table of Content

1	Introduction	4
2	State of Art	5
2.1	Traditional STS Methods	5
2.1.1	Longest Common Substring/Subsequence	6
2.1.2	Greedy String Tiling	6
2.1.3	[Islam and Inkpen, 2008]	6
2.1.4	N-gram Models and Jaccard Similarity Coefficient	7
2.2	Vector Space Model	7
2.2.1	Bag-of-Words and <i>tf-idf</i> Model	8
2.2.2	Latent Semantic Indexing	9
2.2.3	Latent Dirichlet Allocation	9
2.3	Recommendation System	11
2.4	Research Question	11
3	Description of Dataset, Task and Evaluation	12
3.1	Description of Dataset	12
3.1.1	Meta-data and Structure	12
3.1.2	Relatedness and Corpus-Graph	13
3.2	Description of Task	13
3.3	Evaluation Method	15
3.3.1	Effectiveness	16
3.3.2	Efficiency	18
4	Experimental Setup	19
4.1	Theoretical Feasibility and Challenge	19
4.2	Definition and Notions	20
4.3	Architecture of Framework and Experiments Design	21
4.3.1	Preprocessing	21
4.3.2	Model Building and Similarity Computing	22
4.3.3	Model Updating	23
4.4	Experiment Description	23

4.5	Hardware and Software	23
4.6	Dataset	24
5	Experimental Results and Discussion	26
5.1	Analysis of Preprocessing	26
5.2	Effectiveness of STS Models	27
5.3	Efficiency of STS Models	27
5.4	Results of Incremental Updating	27
5.4.1	Effectiveness	27
5.4.2	Efficiency	27
5.5	Error Analysis	27
5.6	Conclusion	27
6	Combination of Approaches	28
7	Conclusion	29
	Bibliography	30

1 Introduction

2 State of Art

In this chapter the common methods to calculate similarity between documents are summarized. The methods are usually categorized into two major classes, which are corpus-based and knowledge-based measures. Corpus-based measures try to calculate the value of similarity between documents using information exclusively extracted from large corpora. Normally, the corpora is used both for generating necessary knowledge (e.g. word2vec) and as input data for training models. Meanwhile, the knowledge derived from other resource, e.g. Wikipedia, New York Times, is drawn into the initializing of building models in knowledge-based measures. In our case, the corpus-based measures is preferred rather than knowledge-based measures. The reasons are briefly explained as follows. Firstly, the existent knowledge-based measures are more suitable in the field of word similarity, e.g. [Jiang and Conrath, 1997], [Strube and Ponzetto, 2006] and [Agirre et al., 2009], but they are relatively weak for dealing with documents or paragraphs. The second viewpoint is that the prior knowledge has less relevance to the target corpus. The importance and meaning of a word or phrase could be different in different resource, so that the knowledge from other resource could not interpret identical words in a specific corpus. Furthermore, the knowledge-based measures have relatively unsatisfactory performance in case of dealing with the considerable number of Out of Vocabulary (OOV) words, whereas exactly compound words and inflection are ubiquitous in German.

In the following subsection, the traditional approaches of semantic text similarity are introduced. After that, a few of important measures of Vector Space Model (VSM) are described. The third part focuses on a short review of existent recommendation systems. The section concludes with a formulation of the research questions.

2.1 Traditional STS Methods

[Bär, 2013] proposes a classification schema which categorizes the measures into *compositional measures* and *non-compositional measures*. In short, *compositional measures* treat documents as a sequence of words and compute document similarity by aggregating pairwise word similarity. *Non-compositional measures* regard a document as an entirety and represent it using a model. The overall document similarity is computed by comparing the representations. One of *compositional measures*, introduced in [Islam and Inkpen, 2008] and three *non-compositional measures*, called LCS, GTS, Jaccard Coefficient respectively, are described as follows.

2.1.1 Longest Common Substring/Subsequence

Longest common substring is based on a basic idea that document is treated as sequence of characters and are compared to each other. The similarity between two texts t_1 and t_2 is:

$$sim_{LCS}(t_1, t_2) = 1 - \frac{l(t_1) + l(t_2) - 2 \cdot l(lcs(t_1, t_2))}{l(t_1) + l(t_2)} \quad (2.1)$$

where the length l of the longest contiguous character sequence lcs is shared between the two texts. Moreover, longest common subsequence overcomes the limitation of continuousness.

The kind of measures have the drawbacks. One of them is that the measures do not utilize any semantic process, while the measures cannot use any optimization method, so that the operating efficiency is in a low level. Assumed there are d documents in the corpora, average \bar{w} words for each document, and average \bar{c} characters for each word, the time complexity of lcs is $O((d \cdot \bar{w} \cdot \bar{c})^2)$. These drawback occur also in GST, which is drawn in next part.

2.1.2 Greedy String Tiling

Greedy string tiling is implemented by [Wise, 1993]. The purpose is to detect plagiarism in programming assignment of students. The algorithm works by searching for matches of a maximum-length marking them and continuing the search ignoring marked tokens. Therefore, this method is relatively insensitive the order of words. It makes more sense for similarity computing than LCS. However, the time complexity is worse than LCS. It will be $O(d^2 \cdot (\bar{w} \cdot \bar{c})^3)$ in the worst case. Taking into consideration of a reasonable efficiency, this method is not used in our experiments.

2.1.3 [Islam and Inkpen, 2008]

[Islam and Inkpen, 2008] introduces an extension of longest common subsequence, that computes not only the scores of lcs (v_1), but also the scores of maximal consecutive longest common subsequence at character 1 (v_2) and at character n (v_3). The overall score of similarity is the weighted mean of the three scores, mathematically, $sim = w_1 v_1 + w_2 v_2 + w_3 v_3$, where $w_1 + w_2 + w_3 = 1$.

The above described three measures regard documents as sequences of characters, so that they are not able to represent the latent relatedness of documents and they are no longer in force when they deal with long (and with multiple semantics) documents. Moreover, considering the operating efficiency these measures make sense only for short information such as title and summary of documents.

2.1.4 N-gram Models and Jaccard Similarity Coefficient

From the viewpoint of n-gram models, the elementary unit of documents is not isolated words but combinations of n words, where n equals 2 or 3 typically. N-gram models provide a basic processing idea, that most measures can utilize n-gram models as a part of the workflow, in order to obtain the (probabilistic) association between a word and $n - 1$ words as predecessor thereof. One simple and intuitionistic measure is Jaccard similarity coefficient [Bank and Cole, 2008]. For two documents $t_1 = \{w_1^1, w_1^2, \dots, w_1^{|t_1|-n+1}\}$, $t_2 = \{w_2^1, w_2^2, \dots, w_2^{|t_2|-n+1}\}$, Jaccard similarity coefficient is defined as the size of the intersection divided by the size of the union. Formally,

$$sim_{jaccard}(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|} = \frac{|t_1 \cap t_2|}{|t_1| + |t_2| - |t_1 \cap t_2|}. \quad (2.2)$$

2.2 Vector Space Model

The idea of VSMs is to represent a document composed of a word sequence of an uncertain or infinite (e.g. data stream) length as a point in a vector space of a finite and determinate high-dimension. Computing the similarity of two documents t_1, t_2 , is accordingly transformed as computing the similarity of the corresponding vectors $\mathbf{v}_1 = (v_1^1, \dots, v_1^s)$, $\mathbf{v}_2 = (v_2^1, \dots, v_2^s)$. The most common method is to compute *cosine* similarity:

$$sim_{cos}(t_1, t_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1| \cdot |\mathbf{v}_2|} = \frac{\sum_{i=1}^s v_1^i \cdot v_2^i}{\sqrt{\sum_{i=1}^s (v_1^i)^2} \cdot \sqrt{\sum_{i=1}^s (v_2^i)^2}} \quad (2.3)$$

The *cosine* similarity of two documents ranges from 0 to 1, when all components thereof are non-negative, e.g. in the tf-idf model, while it will range also from -1 to 1 without the above non-negative requirement, e.g. in topic models. A variant *cosine* similarity is so-called Pearson correlation, where vectors are normalized by subtracting the vector means $\bar{\mathbf{v}}$. Mathematically,

$$sim_{pearson}(t_1, t_2) = \frac{(\mathbf{v}_1 - \bar{\mathbf{v}}) \cdot (\mathbf{v}_2 - \bar{\mathbf{v}})}{|\mathbf{v}_1 - \bar{\mathbf{v}}| \cdot |\mathbf{v}_2 - \bar{\mathbf{v}}|} \quad (2.4)$$

Another measure to indicate the similarity of vectors is to compute the distance, such as Euclidean distance and Manhattan distance. In this kind of similarity measure, vector normalization must be taken

into account. One typical method is to make the norm of vectors as 1, so that documents of different length are treated in the equality.

$$sim_{euclidean}(t_1, t_2) = 1 - \frac{\sqrt{\sum_{i=1}^s (v_1^i - v_2^i)^2}}{\sqrt{\sum_{i=1}^s r^2}}, \quad (2.5)$$

where r is the value range of components of vectors.

VSMs can be classified as term-weighted models and topic-weighted models according to different methods of representing. In following sections we will introduce two common measures of these two types respectively and discuss their advantages and challenges.

2.2.1 Bag-of-Words and *tf-idf* Model

The bag-of-words (BoW) model is a simplest term-weighted vector space model. The vector of a document is composed of the frequency of occurrence of all words in the vocabulary. The dimension of BoW vector therefore is equal to the size of the vocabulary which is normally generated using the entire corpus. The BoW model has a crucial lack that some words occur in the most documents frequently, such as “go”, “buy”, “play” in English. After normalization, these words will occupy the dominating position and the weight of other words will be insignificant. Quite the opposite, words, which occur infrequently in the corpus and relatively repeatedly in a cluster, make more sense in the specific context or topic than the common words.

From this view point, term-frequency-inverse-document-frequency (*tf-idf*) exhibits the importance to provide the meaning or the characteristic in a specific document but not frequency itself of words. The idea contains of two parts. On one side, the metric *tf* indicates how frequent the current word is in the target document. On the other side, the metric *idf* exhibits the scarcity of the word in the entire corpus. Given a document d from the corpus D and an arbitrary word w_i in the vocabulary,

$$tfidf(w_i, d) = tf(w_i, d) \cdot \log_2\left(\frac{|D|}{df(w_i, D)}\right), \quad (2.6)$$

where *tf* is the number of occurrences of w_i in d , and *df* is the number of documents containing w_i .

In both of BoW and *tf-idf* models, the representation of documents is sparse vectors, of which in average more than 90% components are zero, because the vocabulary size is about 10k~30k and each document has typically 300~3k words. Furthermore, ignoring the ordering of words in documents for any measure based on the BoW model will lose information of semantics, e.g. relatedness of words,

causal relationship between sentences. The former shortcoming will be overcome in topic-weighted measures reviewed in the next paragraphs.

2.2.2 Latent Semantic Indexing

The fundamental idea of LSI is dimensionality reduction, i.e. that documents represented in a high-dimensional term-weighted space are converted into a lower dimensional space, so-called latent semantic space. Each component of vector in the latent semantic space can be understood as the weight of a specific topic. [Deerwester et al., 1990] found a way based on linear algebra, called Singular Value Decomposition (SVD), to realize the conversion of vector spaces. Then, [Landauer et al., 1998] applied the idea to compute semantic similarity. The measure, that the SVD also can be applied for computing document similarity, is called Latent Semantic Indexing (LSI).

The standard SVD is $N = U\Sigma V^t$, where Σ is a diagonal matrix, U and V are orthogonal matrices, i.e. $U^t U = V^t V = I$. The sketch of the process of SVD for the corpora is depicted in figure 1

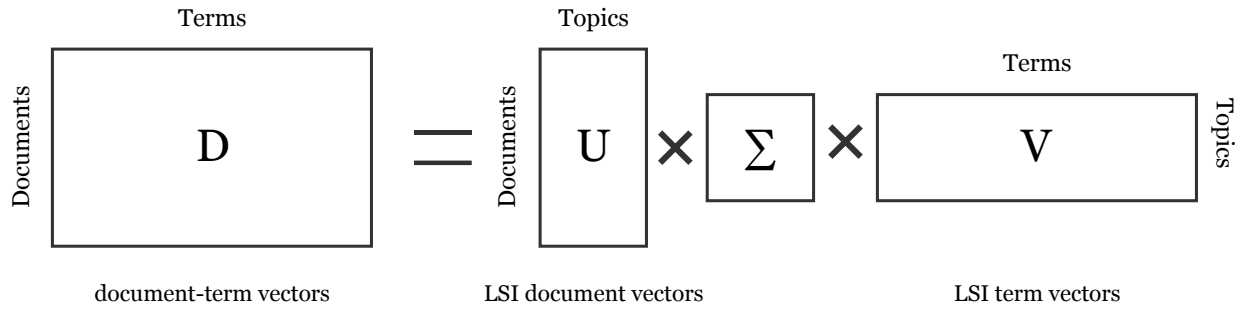


Figure 1: the sketch of the process of LSI using SVD

The LSI model is able to recognize synonyms and reduce the adverse impact of noise of data, so that the documents are profiled more precisely and more stably. However, it has also disadvantages. On the one hand, SVD is an operation with high complexity of runtime and memory. When a huge number of documents are trained or the model needs to update, the cost of SVD is probably intolerable (see section ??). On the other hand, the meaning of the components of V and U cannot be interpreted as a kind of human understanding.

2.2.3 Latent Dirichlet Allocation

LDA was presented as a probabilistic graphical model by [Blei et al., 2003]. LDA is a BoW-based topic model, which can compute the probability distribution of topics for a given document. Moreover, topics can be represented as a series of weighted words by LDA.

The LDA model ([Blei et al., 2003]) is a Bayesian generative model. From the view point of LDA, a document is generated in the following phases:

1. select the topic distribution θ_i of the target document d_i from the prior Dirichlet distribution α ,
2. generate a topic $z_{i,j}$ of j th word in the document d_i from the multinomial topic distribution θ_i using Gibbs sampling,
3. generate a word distribution $\Phi_{z_{i,j}}$ of the topic $z_{i,j}$ from the prior Dirichlet distribution β using Gibbs sampling,
4. generate the word from the multinomial word distribution $\Phi_{z_{i,j}}$.

The progress of generation is illustrated by figure 2.

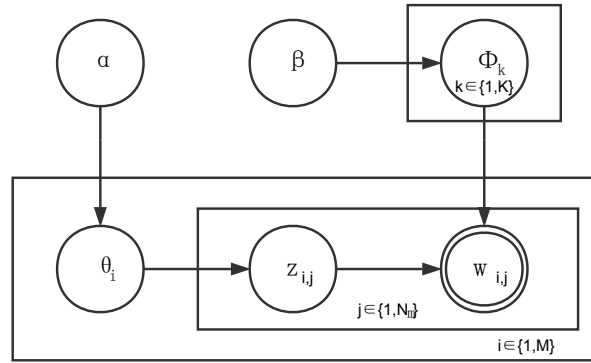


Figure 2: Plate notation of representation of LDA. The dual-line plate refers to the observed variable, i.e. the posterior distribution of words in this case, and the monocoil plates indicate the latent variables, such as topic distribution and word distributions given a topic. The direction of arrows refer the conditional dependency of variables.

The LDA model provides a general idea to build topic model. Since the work of [Blei et al., 2003], subsequent research has explored many variant of LDA. There are four kinds of topic models so far, where are 1) unsupervised and non-hierarchical, 2) unsupervised and hierarchical, 3) supervised and non-hierarchical and 4) supervised and hierarchical. The research and model names are drawn in the table 1 respectively.

Topic Models	non-hierarchical	hierarchical
unsupervised	LDA [Blei et al., 2003] Correlated Topic Model [Blei and Lafferty, 2006]	HLDA [Sivic et al., 2008]
supervised	sLDA [Blei and McAuliffe, 2008] labeled LDA [Ramage et al., 2009]	HSLDA [Perotte et al., 2011]

Table 1: Topic Models Variant of LDA

2.3 Recommendation System

2.4 Research Question

There is a nontrivial way from semantic text similarity to relatedness of documents. We attempt to find a systematic measure to discover related articles by given a target one. The main task is to discover 2~5 candidate of related articles from a large corpora by given a target document. The most of researches have only discussed the degree of similarity between documents or the proportion of topics in each document and in the corpus. However, text similarity does not mean relatedness, neither do a few shared topics of documents. For example, a piece of news reporting the personal financial problems of the president Obama and another discussing making the financial policy of the president Obama are unrelated to each other by human understanding, whereas the STS measures treat them as related articles due to the high score of text similarity. More about this kind of errors will discuss in section ??.

Four major questions have to be answered, so that achieving the relatedness degree of documents can be formulated more properly:

1. [1] Can Semantic Text Similarity including string-based and vector space models methods be used to find related articles? How effective and efficient is the metrics of performance, such as precision and operating time?
2. [2] Taking reducing time and memory usage into account, some documents, which are clearly unrelated to the target, should be filtered before processed by STS measures. Can a filtering method based on data and meta-data be applied to reduce the number candidates that must be checked?
3. [3] Is it useful to combine STS methods with semantic vector space? Does it yield to an improvement in performance or to a significant decrease of runtime?
4. [4] The above introduced methods are unsupervised. Does it lead to an improvement when (semi-) supervised methods are utilized?

3 Description of Dataset, Task and Evaluation

In this section we introduce the target dataset, the structure of articles and the relationship between articles which is regarded as a graph, called corpus-graph. Then, we bring forward the task of our work that a framework or a recommendation system is designed to discover related articles automatically instead of accomplishing the job manually. The final goal is to make the break-even point between the precision of predicting and the operating performance. In the end of the section, we discuss which evaluation methods are chosen and the reason thereof.

3.1 Description of Dataset

Our task concentrates on the corpus **ZEIT ONLINE**. “DIE ZEIT” (eng. “the time”) is a German national weekly newspaper and “ZEIT ONLINE”¹ is the corresponding online representation. More than 300,000 articles, which are released between 1946 and 2014 and marked into 20 different categories, are collected in the corpus. Basically, one or two articles are indicated as recommended reading for each article. In the past, normally, this kind of assignments is completed by editors manually. The motivation of this research is to find an automatic or semiautomatic alternative approach to accomplish the job. Foremost, the elementary information and statistic are useful for a better understanding of the corpus.

3.1.1 Meta-data and Structure

The record of each article consists of a structure of meta-data, which describe the elementary information of the article. The structure is introduced briefly in this section. Then a typical example is depicted in figure 3.

URL the hyperlink as the entry to access the article.

Title a descriptive heading or caption, in which the article is summarized.

Supertitle a label of the article, which indicates a particular theme under the category.

Summary a couple of sentences where is used for arousing the interest of reader and which are the explanation of the background or the summarization of the article.

Author the writer of the article.

Release date the timestamp of publishing the article. The distribution is illustrated in table 2.

Category the category which the article belongs to. The distribution is illustrated in table 3.

¹ ZEIT ONLINE: <http://www.zeit.de/index>

Content the main body of the article.

Keywords a couple of nouns or phrases which are significant for the article and can be used for retrieval. Normally, the vocabulary of all possible keywords is maintained manually.

Related articles the recommendation for extended reading according to the current article. The related articles may have the same topic, similar background, or causal relationship to the current article.



3.1.2 Relatedness and Corpus-Graph

As introduced in the previous section, each article has one or two recommended articles. From the viewpoint of graph theory, the corpus is treated as a undirected graph, in which articles are regarded as vertices and the relationship of recommendation are as edges between the target article and recommended articles. Formally, two articles can be labeled as “related”-by- h to each other, if a path between them exists in the path and the length thereof is shorter than the pre-defined value h (by default: 3). For instance represented in figure 4, article A is related to article B, C, D, E, F, F, G , and unrelated to article H (too far from the target), I, J, K (unreachable from the target).

3.2 Description of Task

In the past, the relationship of relatedness are labeled typically by the author. However, this way has some disadvantages. On the one hand, the job need more human costs. On the other hand, the selection of candidates is limited by the work capacity and knowledge of the author, and these level is not stable. So it is necessary to exploit an automatic framework to discover related articles. The input of the framework is a article represented in a structure described in section 3.1.1, while the output is k (in our case, $k = 2$) articles which are marked as “related” normally with highest score or probability computed by the model. After determining the input and the expected output format, the main task is to train the model with the corpora of historical articles. The training methods are classified into unsupervised and supervised methods. The both kinds of methods are elaborated in section ???. In the scenario of reality, articles come in chronological order and the corpora which is as dataset of candidates has to be updated timely, because articles which are close to each other in time prefer to be more related than articles which are distant in time (see section ??). Furthermore, it is also essential to update the model, in order to improve the precision, while operation of updating is costly and will lead to decrease the operating performance. Another task is, hence, to find an incremental method of updating and the trade-off between the effectiveness (precision) and efficiency (operating performance). The high-level depiction is drawn in figure 5.

URL


www.zeit.de/politik/ausland/2016-06/us-wahl-donald-trump-geldsorgen


CATEGORY

Politik

Gesellschaft

Wirtschaft

Kultur

Wissen

Digital

Campus

Karriere

SUPERTITLE

US-Wahl

TITLE

Die 1,3 Millionen Probleme von Donald Trump

SUMMARY

Die Wahlkampfkasse des republikanischen Präsidentschaftskandidaten ist leer, er aber verbreitet Optimismus. Doch erste Anzeichen von Schwäche sind nicht zu übersehen.

AUTHOR

Von Martin Bialecki

RELEASE DATE

22. Juni 2016, 17:23 Uhr / Quelle: ZEIT ONLINE, dpa / 77 Kommentare

CONTENT

In US-Wahlkämpfen werden mittlere zweistellige Milliardenbeträge umgesetzt. Für maschinenartige Organisationen, für Anzeigen und alle Formen von Kampagnen. Nun ist 2016 aber alles anders, und viel bringt nicht immer viel. Auch die Rolle des Geldes ist vielleicht eine andere, so war es auch bei Jeb Bush, dem ausgeschiedenen republikanischen Bewerber: 130 Millionen hat der Sohn und Bruder früherer Präsidenten für seine Kandidatur verbraucht, dann stieg er erschöpft aus.

KEYWORDS

Schlagworte

USA Präsidentschaftswahl Donald Trump US-Präsident Florida New York

RELATED ARTICLES

LESEN SIE JETZT

US-Präsidentschaftswahl

Wer folgt auf Obama?

23. März 2016

Figure 3: a typical example of the structure and meta-data of articles

year	quantity	proportion (%)	year	quantity	proportion (%)
1946	822	0.27	1981	3432	1.12
1947	658	0.22	1982	2898	0.95
1948	1715	0.56	1983	2849	0.93
1949	1579	0.52	1984	2831	0.93
1950	855	0.28	1985	4363	1.43
1951	662	0.22	1986	4138	1.36
1952	588	0.19	1987	4499	1.47
1953	612	0.20	1988	4328	1.42
1954	540	0.18	1989	4361	1.43
1955	794	0.26	1990	4242	1.39
1956	728	0.24	1991	4476	1.47
1957	660	0.22	1992	4604	1.51
1958	774	0.25	1993	4001	1.31
1959	737	0.24	1994	4498	1.47
1960	1613	0.53	1995	5767	1.89
1961	3517	1.15	1996	5765	1.89
1962	3408	1.12	1997	5510	1.81
1963	3517	1.15	1998	2560	0.84
1964	3984	1.31	1999	5681	1.86
1965	4405	1.44	2000	6904	2.26
1966	4851	1.59	2001	7124	2.33
1967	4662	1.53	2002	4578	1.50
1968	5068	1.66	2003	4475	1.47
1969	5299	1.74	2004	5049	1.65
1970	5293	1.73	2005	5031	1.65
1971	4596	1.51	2006	2782	0.91
1972	5266	1.73	2007	1350	0.44
1973	5155	1.69	2008	6746	2.21
1974	4596	1.51	2009	14836	4.86
1975	3804	1.25	2010	15328	5.02
1976	3616	1.18	2011	14396	4.72
1977	3535	1.16	2012	15419	5.05
1978	3606	1.18	2013	16827	5.51
1979	2969	0.97	2014	6039	1.98
1980	2964	0.97	nan	15	0.00
total		305222	100.00		

Table 2: Release date distribution of the corpus

3.3 Evaluation Method

Effectiveness and efficiency are two main performance indicator. Effectiveness is about doing the right things. particularly in our case, it refers to whether the framework discovers related articles. Efficiency is about doing the things in optimal way, i.e., how much time and memory usage a series of operations to find related articles consumes.

year	quantity	proportion (%)
Politik	80482	26.37
Wirtschaft	57530	18.85
Kultur	55794	18.28
Gesellschaft	27600	9.04
Wissen	26078	8.54
Lebensart	23714	7.77
Reisen	13968	4.58
Sport	7427	2.43
Digital	4267	1.40
Karriere	3352	1.10

year	quantity	proportion (%)
Studium	2551	0.84
Literatur	1384	0.45
Deutschland	369	0.12
Hochschule	360	0.12
Auto	147	0.05
Musik	88	0.03
Gesundheit	53	0.02
Meinung	51	0.02
Kunst	6	0.00
Schule	1	0.00

total 305222 100.00

Table 3: Category distribution of the corpus

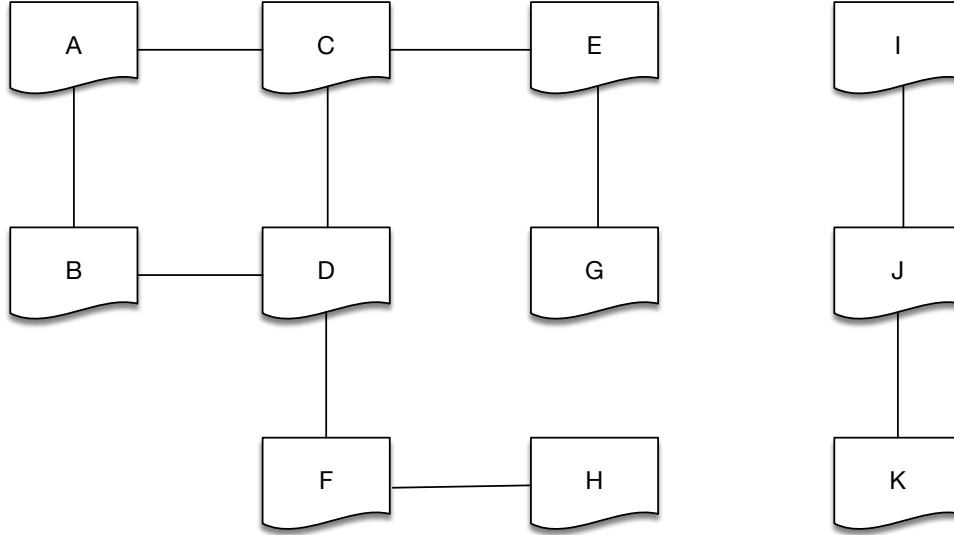


Figure 4: a schematic graph of articles and relationship of relatedness

3.3.1 Effectiveness

The evaluation of effectiveness can be separated into two parts. One of them is to evaluate how well the final results and the other one is to indicate how well the entire ranking of candidates are computed by the framework given the target article. *Precision@k@h* is used for the former and *normalized Discounted Cumulative Gain (nDCG)* is applied for the latter.

Precision@k@h is defined as the ratio of the number of the correct predictions to the number of predictions. Formally, given the corpora $D = \{d_1, d_2, \dots, d_n\}$, a target set $T = \{t_1, t_2, \dots, t_s\}$, the framework

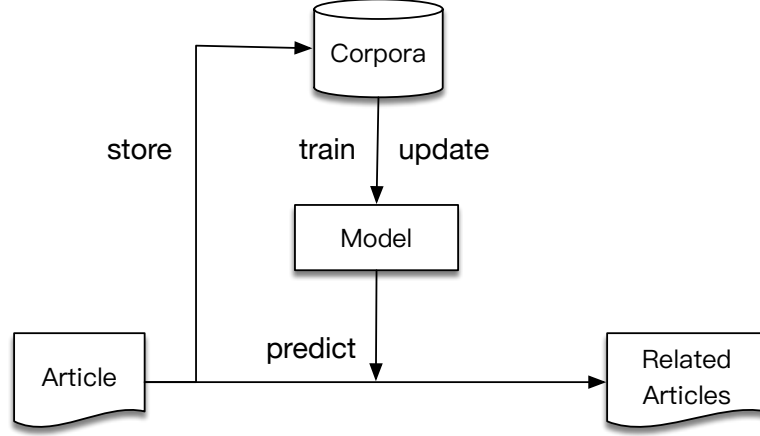


Figure 5: High-level depiction of the framework to discover related articles.

predicts k related articles $P_i = \{d_{i_1}, \dots, d_{i_k}\}$ for each article i in T . Articles, which are at most h -hops apart from the target article, are regarded as the set R_i^h of the true related articles of the article i in T .

$$precision@k@h = \frac{\sum_{i=1}^s |P_i \cap R_i^h|}{k \cdot |T|} \quad (3.1)$$

Unlike $precision@k@h$ which is applied to evaluate the final results of the framework, $nDCG$, which is a measure of ranking quality in the field of information retrieval, focuses to evaluate the relatedness degree of the entire candidates. The framework makes a ranking of all n candidates by the relatedness score or probability for a given target article t . Note that a article d_i is placed at the position i in the ranking. The relatedness degree is defined as 2^{-l} where l is the distance between two articles in the corpus-graph.² We denote the degree between the target t and a candidate d_i by $ref_{d_i}^t$.

The formula to compute $nDCG$ of a target article t is as followed.

$$nDCG_t = \frac{DCG_t}{IDCG_t} = \left(\sum_{i=1}^n \frac{ref_{d_i}^t - 1}{\log_2(i+1)} \right) / \left(\sum_{i=1}^n \frac{ref_{d_i}^t - 1}{\log_2(r_{d_i} + 1)} \right) \quad (3.2)$$

Where r_{d_i} is the position of d_i of the ideal ranking which is generated by ordering by the relatedness degree. The $nDCG$ of the target set is $nDCG = \sum_{t \in T} nDCG_t / s$.

² $l = +\infty$ and the degree is 0, if the articles are unreachable to each other.

3.3.2 Efficiency

The newspaper is quite time-sensitive, so that it is significant that all tasks inclusive of discovery related articles should be completed as soon as possible. The time and memory usage of building model, predicting and updating model should be evaluated respectively. We focus mainly on the divergence between different methods in order to compare their relative merits, whereas the concrete value of time cost and memory usage depends on the performance of hardware and hence we discuss just whether the cost is in a reasonable range. For instance, the operation of predicting should be finished in one minute maximal and it is reasonable and tolerable that building model is completed in one day or even a couple days.

4 Experimental Setup

In this section, we discuss how to design the framework for discovering related articles in the corpus. As described in section 3, each article contains semantic data including title, summary and content, together with meta-data, which consists of category, keywords, release date and the number of words in the texts. The purpose of this section is to evaluate how the methods which are mentioned in section 2 work in effectiveness and efficiency together with different semantic data. Then the assessment of filtering method which reduces the number of candidates with combining the meta-data is analyzed. In other words, this section focuses on finding the answer of question 1 and 2.

In section 4.1 the theoretical feasibility and challenge is discussed. The definition of terms and notations which are used in the rest sections is brought forwards in section 4.2. In section 4.3 we discuss the design of experiments according to the application scenarios and depict the architecture of the framework. Furthermore, the setup of experiments is described in section 4.4, including the setup of dataset given to experimenting and the setup of alternative models.

4.1 Theoretical Feasibility and Challenge

All STS methods and VSM models which are introduced in section 2, are usually applied for computing the semantic similarity between texts. However, our goal is to find related articles for given target articles. Foremost, the difference between the term *similarity* and *relatedness* must be understood. The terms *similarity* and *relatedness* are two separate concepts[Pedersen et al., 2007]. *Similarity* is the measure which indicates how a text looks like another text semantically, while *relatedness* is a more general concept, which contain multiple notions, such as causal relationship, temporal relationship and the relationship shared the identical event or background. We illustrate the difference with two examples.

Example 1 There are two pieces of news. One of them reports the football game of Euro Champions between FC Bayern and Real Madrid in 2014, while the other reports the football game of German Bundesliga between FC Bayern and Dortmund in 2015. Obviously, they are similar to each other semantically, because they share the same topic (football game) and the same subject (FC Bayern). Meanwhile, they are unrelated to each other. The reason is that the two games are held in the different seasons and the different competitions. Exactly, football games is quiet a kind of short-term and event-sensitive events.

Example 2 Case TBD. They are not similar, because they may share quiet a few words and meaning. However, they are related, because the both are about the consequence of TBD. The prediction of relatedness requires knowledge of the background in this case.

From the two examples, we can draw a conclusion, that computing relatedness is much more complicated than computing similarity, because the machine must understand exactly how humankind understands the identities and the differences between two articles as well as the significant degree thereof. However, *similar* documents and *related* documents have a non-negligible intersection normally. Our task is derived from a scenario of reality that two related articles need to be assigned for every current article. Therefore, it is unnecessary to find all possible related articles but it is acceptable to find only a subset of them. From this goal, a way from computing *similarity* to getting *related* articles is feasible.

Certainly, discovery related articles using similarity methods leads to bias. It means, that the framework prefers predicting related articles only with specific characteristics and consequently some articles will never be assigned, e.g. articles which have the identical background with the target. The bias and error analysis are discussed in section 5.

4.2 Definition and Notions

In the following sections, a series of concepts is mentioned frequently. In order to avoid ambiguity and misapprehension, the definition and interpretation of these terms are given in the following list.

Article an instance of a piece of news or report, which contains the complete information or rather texts of title, summary and content as well as meta-data. Denoted as D .

Document a specific component of an article which refers to the string of content without any explicit declaration as well as the string of title or summary when an explicit declaration is given. Denoted as d .

Candidate any article, which needs computing similarity with the target article, within the historic corpus.

Token the elementary semantic unit of a document

Word (in the n -gram models) the elementary semantic unit of a document, which consists of n adjacent tokens. Denoted as w_i .

Vocabulary the set of words which occur in the corpus. In VSMs, the vocabulary contains the words which occur in the corpus at least k times (in our case, $k = 5$), in order to reduce the dimension of document-term-vectors.

Term the unique item of the vocabulary V which is generated by the entire corpus. Denoted as $t_i \in V$.

Related-graph the graph in which the vertices are connected to each other according to the recommendation relationship which is stored in the corpus. If a path with two vertices exists, the

corresponding articles are regarded as related to each other with the length h of the path. Denoted the two articles are related with **hop- h** .

4.3 Architecture of Framework and Experiments Design

The high-level architecture of the framework was already depicted in figure 5. In this section, we explain how the framework predicts related articles and improves itself.

The framework consists of four phases of preprocessing, model building, similarity computing and model updating. The architecture of the framework is illustrated in figure 6.

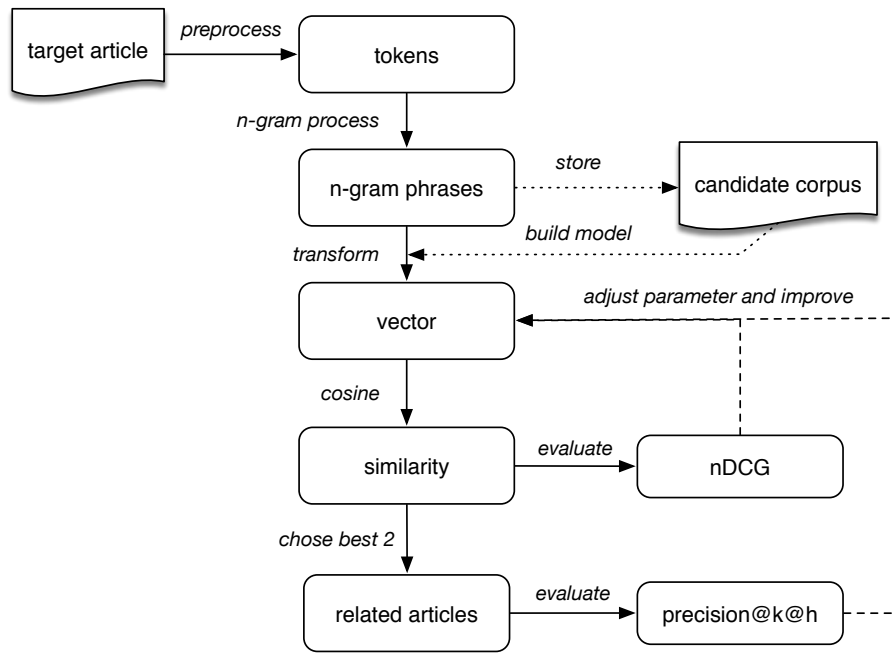


Figure 6: The architecture of the framework to discover related articles.

4.3.1 Preprocessing

The original data of title, summary and content is stored in the strings in HTML format. After all markups are removed and all characters are converted to small case, preprocessing separates the strings into the respective sequences of tokens. Four common methods are explained as follows:

SPlit The string is splitted into a sequence of tokens with the special characters (whitespace, hyphen and punctuations).

StEm Tokens, which are the result from the method *SP* are replaced by the respective stems. For example, “book” is the stem of “books”, “booking” and “booked”. The advantage of stemming is

to reduce the vocabulary size and decrease computational complexity and avoid overfitting of building semantic models. However, stemming causes the more serious problem of polysemy.

STop After *SP*, the stopwords, which are the most common words in a language, such as pronouns and prepositions are filtered out from the sequence.

Stem+Stop The data is handled by both of *SP* and *SE*.

Preprocessing contains a sub-component, called n-gram processing. In our case, unigram, bigram and trigram are applied. In total, we have $4 \times 3 = 12$ alternative preprocessing methods. After preprocessing, the strings are represented as the respective sequences which the semantic models are able to deal with directly.

4.3.2 Model Building and Similarity Computing

The component of model building is meaningful exclusive for VSMs. The approaches are quite different according to different models. The detailed methods are described in the following list respectively.

BoW BoW is the simplest and most basic model in VSMs. Each document is represented as a vector in which the weight of each dimension is the occurrence frequency of the respective term. In order to reduce the dimension of vectors, the terms which occur in the corpus at least k times (in our case $k = 5$) remain in the vocabulary. The BoW vector of a given document is independent of the rest corpus, so long as the vocabulary is constant.

tfidf Tfidf is the model based on BoW, but the vector is relevant to the corpus. The tool to build tfidf model and the following models is *gensim*³.

LSI and LDA We build the models LSI and LDA based on tfidf. Both models are capable to reduce the dimension of vectors from the vocabulary size to the number of topics. It is therefore necessary to determine the optimal number of topics, in order to make balance between the precision and computational performance.

In VSMs, each document is transformed as a vector and the semantic similarity to other documents is the *cosine* similarity between the vector of the target document and the vector of candidate documents. The articles in the historic corpus which are the most **two** similar to the given target article are predicted as the related articles.

³ *gensim* is a NLP tool, which focuses on topic modeling and retrieving semantically similar documents [Řehůřek and Sojka, 2010]. Official website: <https://radimrehurek.com/gensim/>

4.3.3 Model Updating

In the application scenario of reality, the historic corpus is not constant but increases over time. Once a article is ready to publish, the framework should select two articles with the greatest relatedness to the current article from the corpus. After the article is released, it should be stored into the historic corpus immediately and becomes the candidate for a related article to future articles. Therefore, the amount of the corpus will increase throughout. In the mean time, the characteristics of the corpus, for example, the topic distribution and the occurrence of new popular words, is rearranged over time. It is necessary thereby to update the model incrementally, in order to be able to reflect the recent situation of the corpus in time. However, the computational time cost of model updating must be taken into account. Accordingly, one of the research tasks is to find the trade-off between delay of updating and the impact on the performance of the effectiveness.

4.4 Experiment Description

TBD

We design two experiments to evaluate the framework. The first experiment focuses on the intrinsic performance of the different STS models with the different preprocessing methods. The intrinsic performance means the performance including effectiveness and efficiency that The training corpus and the testing dataset are constant in the period of execution of the experiment. The purpose of the second experiment is to evaluate the fluctuate of performance when the size of the training dataset increases and the models need updating incrementally. In this case, an article is as the testing data at first and then is stored in the training dataset as the training data to update the models.

4.5 Hardware and Software

The framework is set on a server running Linux system. The detailed information of the server is drawn in table 4

CPU	Intel Xeon CPU E5420 (8 cores) @ 2.50GHz
RAM	32GB
Disk	3TB HDD
System	Debian GNU/Linux 7.7, 64 bits
Runtime	Python 3.4
DBMS	MongoDB
Persistance	HDF5, gzip compressed
Externel Packages	NLTK, gensim, scikit-learn

Table 4: General view of the hardware, system and software in use at major.

4.6 Dataset

We setup the training and testing datasets from the ZEIT-corpus. Articles, which have no related articles or no *title*, or whose content is less than 1000 characters, are filtered. Furthermore, articles which belong to a weak category ⁴ or which were released before 2009 are removed from the corpus. We have 75908 articles in 7 categories. The category distribution is drawn in table 5 and the release date distribution is in table 6

category	quantity	proportion (%)
Politik	26071	34.35
Wirtschaft	12531	16.51
Kultur	8584	11.31
Gesellschaft	7646	10.07
Wissen	5273	6.95
Sport	4993	6.58
Digital	3887	5.12
Reisen	2199	2.90
Karriere	2169	2.86
Studium	1570	2.07
Lebensart	985	1.30
total	75908	100.00

Table 5: Category distribution of ZEIT corpus after filtering unsatisfied articles

year	quantity	proportion (%)
2009	12628	16.64
2010	14716	19.39
2011	13970	18.40
2012	14583	19.21
2013	14941	19.68
2014	5070	6.68
total	75908	100.00

Table 6: Release date distribution of ZEIT corpus after filtering unsatisfied articles

Experiment 1 2000 articles are selected randomly from the corpus as the testing dataset, and the others are as the history data to train the models.

Experiment 2 The articles in corpus are sorted by the *release date*, so that the real-world scenario can be simulated. The articles, which were published before 2013, are as the training data to initialize models. There are two phases to deal with each target article, that are predicting related articles from the historical corpus and updating the model incrementally. Then the

⁴ Weak category: the amount of articles in this category is fewer than 1% of the corpus size

supervised methods make use of the scores from the unsupervised methods to compute the integrated scores and discover the related articles for each target article.

5 Experimental Results and Discussion

The experimental results are described, analyzed and concluded in this section. In section 5.1 the statistics of the corpus after preprocessing is showed briefly. Section 5.2 focuses on the effectiveness of the different STS models, i.e. how precise the STS models predicts an articles as related for a given target article. Furthermore, the operational performance of the STS models is reported in section 5.3. The above three parts concentrate on the first experiment, which is mentioned in section 4.4. On the next step, the results of the second experiment are reported in section 5.4. After reporting the results of the both experiments, we discuss the most severe types of errors that an articles which is virtually unrelated is predicted as related for a target article or a cluster of articles which are properly related are never or rarely predicted as related. In conclusion, we summarize the consequence of selection of the STS models and the challenge of the current work.

5.1 Analysis of Preprocessing

The input data to feed the preprocessing methods is one of three semantic components of an article: *title*, *summary* and *content*. After preprocessing the text in raw string format is converted to a sequence of phrases which consist of n words. For the n-gram model with different value of $n = 1, 2, 3$, we have three different representation forms for every data source, i.e. *title*, *summary* or *content*. A vocabulary is generated from all by preprocessing methods converted data for each representation form of each data source. The size of vocabularies indicates the dimension of document-term vectors as well as impacts the complexity and precision of similarity computing. The advantages of a smaller vocabulary are to decrease the complexity of operations of vectors and to avoid the overfitting of the VSMs and the primary disadvantage is that the semantic information is partly dropped due to the ambiguity of tokens. On the other hand, a larger vocabulary do not lose information, but leads to higher cost of time and memory usage and weaker semantic relevance that the similar meaning is more likely to have quiet different representations. In the n-gram models ($n > 1$), many phrases occur quiet rarely or even only one time in the whole corpus. Figure ??, which depicts a representative sample the distribution of the occurrence frequency, indicates that only a small quantity of tokens occur repetitively in the corpus (DATA). Taking into account efficiency, the tokens which appear less than 5 times can be removed from the vocabulary. We denote the original vocabularies as *full* vocabularies and the reduced vocabularies as *common* vocabularies.

Figure ?? shows the comparison between the size of *full* and *common* vocabularies which are generated by the preprocessing methods including *SP*, *SE*, *ST* and *SS* in the different n-gram models ($n = 1, 2, 3$) of the three kinds of data source containing *content*, *title* and *summary*. In the unigram model, *SS* always

generates the both smallest *full* and *common* vocabularies which are only % – % size of the vocabularies generated by *SP*. *SE* and *SS* produce the smallest *full* vocabularies as well as *ST* and *SP* produce the largest in bigram and trigram respectively. On the other side, the *SP*

The *full* vocabularies in bigram and in trigram is bigger than $10\times$ and $15\times \sim 40\times$ in unigram respectively.

5.2 Effectiveness of STS Models

5.3 Efficiency of STS Models

5.4 Results of Incremental Updating

5.4.1 Effectiveness

5.4.2 Efficiency

5.5 Error Analysis

5.6 Conclusion

6 Combination of Approaches

7 Conclusion

References

- [Agirre et al., 2009] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. (2009). A Study on Similarity and Relatedness using Distributional and WordNet-based Approaches.
- [Bank and Cole, 2008] Bank, J. and Cole, B. (2008). Calculating the jaccard similarity coefficient with map reduce for entity pairs in wikipedia. *Wikipedia Similarity Team*.
- [Bär, 2013] Bär, D. (2013). A composite model for computing similarity between texts.
- [Blei and Lafferty, 2006] Blei, D. and Lafferty, J. (2006). Correlated topic models. *Advances in neural information processing systems*, 18:147.
- [Blei and McAuliffe, 2008] Blei, D. M. and McAuliffe, J. D. (2008). Supervised topic models. pages 1–8.
- [Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation.
- [Deerwester et al., 1990] Deerwester, S., Dumals, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis.
- [Islam and Inkpen, 2008] Islam, A. and Inkpen, D. (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):10.
- [Jiang and Conrath, 1997] Jiang, J. J. and Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.
- [Landauer et al., 1998] Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.
- [Pedersen et al., 2007] Pedersen, T., Pakhomov, S. V., Patwardhan, S., and Chute, C. G. (2007). Measures of semantic similarity and relatedness in the biomedical domain. *Journal of biomedical informatics*, 40(3):288–299.
- [Perotte et al., 2011] Perotte, A. J., Wood, F., Elhadad, N., and Bartlett, N. (2011). Hierarchically supervised latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 2609–2617.
- [Ramage et al., 2009] Ramage, D., Hall, D., Nallapati, R., and Manning, C. D. (2009). Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics.

-
- [Řehůřek and Sojka, 2010] Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. <http://is.muni.cz/publication/884893/en>.
- [Sivic et al., 2008] Sivic, J., Russell, B. C., Zisserman, A., Freeman, W. T., and Efros, A. A. (2008). Unsupervised discovery of visual object class hierarchies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- [Strube and Ponzetto, 2006] Strube, M. and Ponzetto, S. P. (2006). Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, volume 6, pages 1419–1424.
- [Wise, 1993] Wise, M. J. (1993). String similarity via greedy string tiling and running karp-rabin matching.