AMBIGUITY-AWARE DOCUMENT SIMILARITY

Fabrizio Caruso
Fabrizio_Caruso@hotmail.com
Neodata Group

Giovanni Giuffrida
giovanni.giuffrida@dmi.unict.it
Dept of Social Science, University of Catania

Diego Reforgiato
diego.reforgiato@neodatagroup.com
Neodata Group

Giuseppe Tribulato
gtsystem@gmail.com

Calogero Zarba
Neodata Intelligence
calogero.zarba@neodatagroup.com

ABSTRACT

In recent years, great advances have been made in the speed, accuracy, and coverage of automatic word sense disambiguator systems that, given a word appearing in a certain context, can identify the sense of that word. In this paper we consider the problem of deciding whether same words contained in different documents are related to the same meaning or are homonyms. Our goal is to improve the estimate of the similarity of documents in which some words may be used with different meanings. We present three new strategies for solving this problem, which are used to filter out homonyms from the similarity computation. Two of them are intrinsically non-semantic, whereas the other one has a semantic flavor and can also be applied to word sense disambiguation. The three strategies have been embedded in an article document recommendation system that one of the most important Italian ad-serving companies offers to its customers.

KEYWORDS

Disambiguation, Recommendation Systems, Term-Frequency Inverse-Document-Frequency, Online Newspapers, Text Mining

1. INTRODUCTION

When reading newspapers or documents, people are usually unaware of the ambiguities in the language they use: by using document context and their knowledge of the world they can resolve them very quickly. When we think about how computers may tackle this problem we realize how hard it is to solve it algorithmically.

First of all, let us define what ambiguity is. Ambiguityis the property of being ambiguous (homonyms); a word, term, notation, sign, symbol, phrase, sentence, or any other form used for communication, is called ambiguous if it can be interpreted in more than one way. Ambiguity is different from vagueness, which arises when the boundaries of meaning are indistinct. Ambiguity is context-dependent: the same linguistic item (be it a word, phrase, or sentence) may be ambiguous in one context and unambiguous in another context. In general, something is ambiguous when it has more than one meaning.When a single word is associated with multiple senses we have lexical ambiguity. Almost any word has more than one meaning. For example, words like "wound", "produce", "lead", "desert" have multiple meanings.

In this paper we are interested in resolving this form of ambiguity. Other forms of ambiguity are syntactic ambiguity, in which a sentence can be parsed in different ways (e.g., "He ate the cookies on the couch"), and semantic ambiguity, in which a word may have different meanings when used in an informal or idiomatic expression.

For various applications, such as information retrieval or machine translation, it is important to be able to distinguish between the different senses of a word. We address the problem of deciding whether two documents possibly containing ambiguous words talk about the same topic or not. Documents are represented using the bag-of-words model and the term frequency-inverse document frequency metric. In this model, one can determine whether two documents treat the same topic by computing the cosine similarity of the vectors representing the documents.

We present three disambiguation algorithms (which have been included in an article recommendation system) that aim at discerning whether a word shared by two documents is used with the same meaning or with different meanings in the two documents. We remark that our focus is not the investigation of the meaning of the words. Our goal is to improve the estimate of the similarity between documents by heuristically taking out from the similarity computation most of the words that are probably used with different meanings in different documents. To the best of our knowledge, the application of techniques to detect ambiguities for article recommendation is the first in literature.

This paper is organized as follows: Section 2 presents an overview and a literature survey of the problem of word sense disambiguation. Section 3 introduces the term frequency-inverse document frequency metric (tfidf) used throughout the pa-per. Section 4 presents some sense disambiguation issues related to the tfidf we had to face. Section 5 describes the multi-language algorithms we propose. Section 6 depicts the article recommendation system we have used and some results from real data. Finally, Section 7 contains some final remarks on the presented findings.

2. RELATED WORK

Semantic relations (e.g., "has-part", "is-a", etc.) have been used to measure the semantic distance between words, which can be used to disambiguate the meaning of words in texts. Also semantic fields (set of words grouped by the same topic) have proved to be very useful for the disambiguation task. Therefore, with a tool like WordNet [13] and MultiWordNet [5] we have the possibility to work on several problems of computational linguistics for many different languages. Any algorithm created for a language is usually generalizable for all the other languages. WordNet has been widely used to solve a large number of problems related to text mining as document clustering [8]. Below we report some past work where Word-Net and MultiWordNet have been used in approaching the problem of word sense disambiguation (WSD).

The earliest attempts to use WordNet in word sense disambiguation were in the field of information retrieval. Fragos et al. [4] used a dictionary to disambiguate a specific word appearing in a context. Sense definitions of the specific word, “syn-set" definitions, the “is-a" relation, and definitions of the context features (words in the same sentence) are retrieved from the WordNet database and used as an input for a disambiguation algorithm. In [11] the authors used WordNet for WSD. Unlike many others approaches on that area, it exploits the structure of WordNet in an indirect manner. To disambiguate the words it measures the semantic similarity of the words glosses. The similarity is calculated using the SynPath algorithm. Its essence is the replacement of each word by a sequence of WordNet synset identifiers that describe related concepts. To measure the similarity of such sequence the standard tfidf formula is used. In [6] authors deal with the problem of providing users with cross-language recommendations by comparing two different content-based techniques: the first one relies on a knowledge-based word sense disambiguation algorithm that uses MultiWordNet as sense inventory, while the latter is based on the so-called distributional hypothesis and exploits a dimensionality reduction technique called Random Indexing in order to build language-independent user profiles. In [12] the authors propose a method to find Near-Synonyms and Similar-Looking (NSSL) words and designed three experiments to investigate whether NSSL matching exercises could increase Chinese EFL learners' awareness of NSSL words.

Chen et al [3] proposed a novel idea of combining WordNet and ConceptNet for WSD. First, they present a novel method to automatically disambiguate the concepts in ConceptNet; and then they enrich WordNet with large amounts of semantic relations from the disambiguated ConceptNet for WSD.

One of the word sense disambiguation algorithms described in this paper (the one which uses MultiWordNet) builds upon a recent approach, presented in [1, 2, 6], where a method for solving the semantic ambiguity of all words contained in a text is presented. The authors propose a hybrid WSD algorithm that combines a knowledge-based WSD algorithm, called JIGSAW, which they designed to work by exploiting WordNet-like dictionaries as sense repository, with a supervised machine learning algorithm (K-Nearest Neighbor classifier). WordNet-like dictionaries combine a dictionary with structured semantic network, supplying definitions for the different senses and defining groups of synonymous words by means of synsets, which represent distinct lexical concepts.

MultiWordNet

MultiWordNet [5, 7] is a multilingual lexical database developed at "Fondazione Bruno Kessler" in which the Italian WordNet is strictly aligned with Princeton WordNet 1.6 [13]. The current version includes around 44,400 Italian lemmas organized into 35,400 synsets which are aligned, whenever possible, with their corresponding English Princeton synsets. The MultiWordNet database can be freely browsed through its online interface, and is distributed both for research and commercial use [5]. The Italian synsets are created in correspondence with the Princeton WordNet synsets, whenever possible, and semantic relations are imported from the corresponding English synsets; i.e., it is assumed that if there are two synsets in the Princeton WordNet and a relation holding between them, the same relation holds between the corresponding Italian synsets. While the project stresses the usefulness of a strict alignment between wordnets of different languages, the multilingual hierarchy implemented is able to represent true lexical idiosyncrasies between languages, such as lexical gaps and denotation differences.

The information contained in the database can be browsed through the MultiWordNet online browser, which facilitates the comparison of the lexica of the aligned languages.

Synsets are the most important units in MultiWordNet. Here is an example of an Italian synset for the word "computer": “elaboratore”, “computer”, “cervello elettronico”, “calcolatore”. Important information is attached to synsets, such as semantic fields and semantic relations. The semantic field describes the topic of a synset. For example, the above synset belongs to the "Computer Science" semantic field.

3. TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

We compute the similarity between two documents using the well-known bag-of-words model [10]. This model does not consider the ordering of the words in a document. For example, the two sentences "John is quicker than Mary" and "Mary is quicker than John" have the same representation.

In the following, *W* is the set of all words, and *A* is the set of the documents. We assume that both *W* and *A* are finite. In an actual implementation, usually the set of all words does not contain the so-called stop words, which are frequent words with a too general meaning, such as “and”, “is”, “the”, etc. Moreover, words may optionally be stemmed using, for instance, Porter’s stemming algorithm [9].

Let *a* be a document, and let *w* be a word. The term frequency *tf (w, a)* of *w* in *a* is the number of occurrences of *w* in *a*. The raw term frequency does not give enough information to compare two documents. The key point is to consider rare terms.

Let *w* be a word. The inverse document frequency *idf (w, A)* of *w* is given by *idf(w, A)* = $\log\frac{|A|}{1+|A_w|}$, where A_w is the set of documents containing the word *w*. Let *w* be a word, and let *a* be a document. For sake of simplicity we will consider *A* an implicit argument in the subsequent definitions. The term frequency-inverse document frequency *tfidf (w, a)* of *w* in *a* is given by *tfidf (w, a) = tf (w, a)* $\times$ *idf (w)*.

Given a document *a*, we represent its content with a function $f_a : W \rightarrow [0, 1]$. More precisely, we let

$$f_a(w) = f'_a(w) / |f'_a|$$

where

$$f'_a(w) = tfidf(w,a); \qquad |f'_a| = \sqrt{\sum_{w' \in W} \left[f'_a(w')\right]^2}$$

Note that this representation does not consider the ordering of words in documents. Moreover, this representation gives more importance to words that occur frequently within a document (the term frequency part), and to rare words that occur in few documents (the inverse document frequency part). We can also think of f_a and f_a' as vectors whose i-th component corresponds to the i-th word *w* in *W*, and $|f'_a|$ as the Euclidean norm of f'_a.

Given two documents *a, b* the cosine similarity $\sigma(a, b)$ of *a* and *b* is defined as the scalar product of f_a and f_b:

$$\sigma(a,b) = f_a \times f_b = \sum_{w \in W} f_a(w) \times f_b(w)$$

Note that two documents are similar when they have many words in common. In particular, the cosine similarity is higher when the words that the two documents have in common have a high term-frequency and a high inverse-document frequency. Other similarity measures may be considered such as the Pearson correlation index (equivalent to the cosine similarity if the average value is assumed to be zero).

4. Recommending News Documents Using the TFIDF Metric In An Article Recommendation System

In this section we describe how the article recommendation system, (see Section 6 for more details) uses the cosine similarity with the *tfidf* metric to compare documents.

Each time a new document *d* is processed, its similarity with respect to all other documents is computed. For each document *d*, we keep in memory its *k*-nearest-neighbor document list according to the cosine similarity: the first entry contains the most similar document to *d* whereas the last entry contains the least similar. However, it may happen that, for a given document *d*, some of the *k* documents entries are not related to *d* at all. This event may happen when a word with high *tfidf* value has different meanings in different documents. In this case we should declare the word ambiguous and take it out of the similarity computation.
Therefore, we need to perform a word disambiguation task between a given document *d* and each document entry in its *k*-nearest-neighbor document list. Formally, we need to make sure that in the pairs (d, d_1), (d, d_2),…, (d, d_k) no ambiguous words with a high *tfidf* value are used for the computation of the similarity. Ambiguous words with low *tfidf* value do not affect significantly the similarity of a given document pair (d, d_i). Hence we do not need to take them into account.
Given a document pair (d, d_i), we first multiply the *tfidf* values of words occurring in both documents and sort the resulting values in decreasing order. Then, we consider only the resulting terms with a value greater than a fixed threshold *imp_weight*. We will refer to such terms as the *imp_words* set. For the documents we have taken under consideration, the cardinality of *imp_words* ranged between five and ten.

As discussed above, the intuition behind this is that words with a high *tfidf* value highly contribute to the overall score of the similarity between the two documents, and, consequently, highly contribute to the creation of the *k*-nearest-neighbor list.

If a word in *imp_words* is declared ambiguous for a pair of documents (d, d_i), we remove it from further computation of the similarity of that pair. If the value associated to the ambiguous word is high enough, a new computation of *k*-nearest documents of *d* would shift the document di to a more distant position in the nearest-neighbor list, that is, d_i would be declared less similar to *d* than it was previously.

5. The Disambiguation Strategies

Given two documents d_1 and d_2, and a word *w* in the *imp_words* set, we want to decide whether the word *w* is ambiguous for d_1 and d_2. To accomplish this task, we have developed three strategies: (i) the uppercase words strategy, (ii) the word pair frequency strategy, and (iii) the MultiWordNet knowledge strategy. Each strategy consists of an algorithm returning one of three possible values: ambiguous, unambiguous, or undecided. The last value is returned when the algorithm is not able to decide whether the word *w* is ambiguous or unambiguous for the two documents d_1 and d_2.

Uppercase words strategy (UWS)

The uppercase words strategy handles words that are in uppercase, that is, those words denoting named entities such as persons, companies or locations. The idea is that there is ambiguity if an uppercase word *w* denotes two different entities in the two documents. This can be checked by inspecting the other uppercase words near *w* in the two documents.

Let d_1 and d_2 be two documents both containing a word *w*. The word *w* may be in uppercase in at least one of the two documents. Algorithm 1 attempts to decide whether *w* is ambiguous with respect to d_1 and d_2 using the upper case word strategy.

```
function UWS($d_1$, $d_2$, $w$)
    if $w$ is uppercase in one document and lowercase in the other then
        return ambiguous
    else if $w$ is lowercase in both documents then
        return unambiguous
    else //At this point, $w$ is uppercase in both documents
        if there are uppercase words near $w$ in both documents then
            if these uppercase words are the same then
                return unambiguous
            else
                return ambiguous
        else
            return undecided
```

Algorithm 1: Upper case word strategy.

Examples

1. Let d_1 be a document about an Italian company whose name is "Guru". Let also d_2 be a document where the common noun "guru" is used. Since the word "guru" is used in uppercase in d_1 and in lowercase in d_2, the uppercase word strategy declares the word "guru" as ambiguous for d_1 and d_2.
2. Let d_1 be a document about "Carlos Santana", a famous Mexican guitarist. Let also d_2 be a document about "Mario Alberto Santana", a famous Italian soccer player. In this case, the uppercase word strategy declares the word "Santana" as ambiguous for d_1 and d_2.

Word pair frequency strategy (WPFS)

The word pair frequency strategy uses two-word expressions, or word pairs, to try to understand when a word can be classified as ambiguous. The idea behind this strategy is that if a word is used in the same expression within two different documents, then the two documents refer to the same concept.

Given our corpus of documents, we create a set *WP* of commonly occurring word pairs. A word pair, denoted as $w_1|w_2$, is an expression of two words separated by a space. A word pair $w_1|w_2$ belongs to the set *WP* if and only if (i) the word w_i is not a stop word, for $i = 1, 2$, (ii) the word w_i occurs in the corpus at least a predefined minimum number of times *supp*, for $i = 1,2$, and (iii) the expression $w_1|w_2$ occurs in the corpus at least a predefined minimum number of times *min_support*.

Let d_1 and d_2 be two documents both containing a word *w*. Algorithm 3 attempts to decide whether *w* is ambiguous for d_1 and d_2 using the word pair frequency strategy.

```
function WPFS($d_1$, $d_2$, $w$)
    if $w|w'$ □ $WP$□$d_1$□$d_2$, for some word $w'$ then
        return unambiguous
    else if $w'|w$□ $WP$□$d_1$□$d_2$, for some word $w'$ then
        return unambiguous
    else if $w|w'$□ $WP$□$d_1$ and $w|w''$□ $WP$□$d_2$, for some words $w'$, $w''$ then
        return ambiguous
    else if $w'|w$□ $WP$□$d_1$ and $w''|w$□ $WP$□$d_2$, for some words $w'$, $w''$ then
        return ambiguous
    else
        return undecided
```

Algorithm 2: Word pair frequency strategy.

Example

Let d_1 be a document containing the expression “lodo Alfano”, and let d_2 be a document containing the expression “lodo Schifani”. Assume that both expressions are present in our dictionary of commonly recurrent expressions.

In this case, the word pair frequency strategy declares that the word “lodo” id ambiguous for d_1 and d_2.

MultiWordNet knowledge strategy (MKS)

The third strategy exploits the MultiWordNet semantic network. The idea is that, in order to disambiguate a word w with respect to two documents d_1 and d_2, we look at the semantic fields associated to the senses of the words near w in the two documents. If the word w is used in one document in a context entailing a set of semantic fields which is different from the ones entailed by the other documents, it means that the word w is ambiguous.

Let d_1 and d_2 be two documents both containing a word w. Algorithm 3 attempts to decide whether w is ambiguous for d_1 and d_2 using the MultiWordNet knowledge strategy.

```
function MKS($d_1$, $d_2$, $w$)
    $v_1$ = BuildContextVector($d_1$,$w$)
    $v_2$ = BuildContextVector($d_2$,$w$)
    $s$ = □($v_1$, $v_2$)    // □ denotes the cosine similarity function
    if s <min_mw then
        return ambiguous
    else if s <max_mw then
        return undecided
    else
        return unambiguous
```

```
function BuildContextVector($d$, $w$)
    Let $v$ be an empty vector
    for each word $w'$ □ $w$ belonging to any sentence in $d$ containing $w$ do
        for each sense $s$ associated to the word $w'$ do
```

```
15:             Let $f$ be the semantic field of the sense $s$
16:             **if** $f$ is the Factotum semantic field **then**
                    v[f] = v[f] + 0.1 * $tfidf(w',d)$
                **else**
                    v[f] = v[f] + $tfidf(w',d)$
        Normalize $v$ so that its norm is 1
        **return** v
```

Algorithm 3: MultiWordNet knowledge strategy.

Example

Let d_1 be a document containing the following sentence, where the numbers given are the *tfidf* values of the words in the document d_1.

The	virus	infected	the	computer
	0.15	0.1		0.2

Note that there is no *tfidf* value associated with the word "the", because this is a stop word.

Let also d_2 be a document containing the following sentence.

The	HIV	virus	is	the	cause	of	AIDS
	0.05	0.08			0.07		0.1

Using the information from MultiWordNet, we obtain the following vectors of semantic fields for d_1 and d_2.

Factotum	Computer science
0.196	0.981

Factotum	Biology	Law	Sociology	Medicine
0.044	0.443	0.089	0.089	0.886

The cosine similarity of the two vectors is 0.196*0.044 = 0.009. Assuming a minimum threshold value of 0.2, the MultiWordnet knowledge strategy declares that the word "virus" is ambiguous for d_1 and d_2

6. ARTICLE RECOMMENDATION SYSTEM

We have developed an article recommendation system, which works on Italian documents and includes the disambiguation strategies described above. The considered text documents were the Italian news published daily by one of the most important Italian newspapers which contains news about politics, environment, weather, technology, world events, finance, sport, travel, etc. A dedicated crawler continuously extracts the information about each new document (title, text, url, section, date, etc.). The extracted documents are first cleaned of all HTML entities, spell-checked and, finally, stored in a highly optimized database that we have designed on top of the PostgreSQL system. The database consists of different tables containing extensive information about the extracted documents (document id, title, content, url, date, category); moreover, for each word there is a table containing its frequency in the dataset. On average, 50 new documents per day are published, and, consequently, extracted by our crawler and stored into the database. The collected documents at the time of the current study were about 150,000. Given a document *d*, our article recommendation system returns a list of the most similar documents to *d* filtering out any potential ambiguities.

Dates	***Impressions***	***Clicks***	***CTR***
2011-09-16	96142	2640	2.745%
2011-09-17	58918	1949	3.307%
2011-09-18	58365	2193	3.757%
2011-09-19	80596	2483	3.080%
2011-09-20	86306	2271	2.631%
2011-09-21	80190	2043	2.547%
2011-09-22	98253	2307	2.348%
2011-09-23	79597	1984	2.492%
2011-09-24	49353	1821	3.689%
2011-09-25	46284	1798	3.884%

Table 1: Impressions, Click and CTR for recommended articles between 2011-09-16 and 2011-09-25.

Results on real data

In the Table 1 we have collected some results on real data. Once a user reads an article, a list of suggestions is shown at the end of the article. We have counted the number of impressions and clicks on recommended articles in one of the sites that use our article recommendation system in the period between 2011-09-16 and 2011-09-25. There is an impression for an article *a*, whenever the article *a* appears in the list of suggestions. There is a click for an article *a*, whenever a suggested article *a* is clicked by the user. The data in Table 6.1 show a high click-through rate, i.e., the ratio between clicks and impression, which is about 3%. This means that a large number of articles are read as a consequence of the suggestions produced by our article recommendation system.

CONCLUSION

This paper presents a novel way to calculate document similarity using the term frequency-inverse document frequency metric improved by three different disambiguation strategies. In particular, we have proposed a strategy, which exploits information about word case, a strategy which uses information about frequency of multi-word expressions, and one more strategy which uses MultiWordNet semantic information. These disambiguation strategies can be embedded in a system for any language (in particular, the Italian language for the results and experiments presented in this paper). They have improved the precision of the article recommendation system where they have been embedded.

REFERENCES

[1] P. Basile, M. de Gemmis, A. L. Gentile, P. Lops, G. Semeraro, The jigsaw algorithm for word sense disambiguation and semantic indexing of documents, in: Proceedings of AI*IA07, 10th Congress of Italian Association of Artificial Intelligence. Roma Italy, pp. 314-325.

[2] P. Basile, M. de Gemmis, P. Lops, G. Semeraro, Combining knowledge-based methods and supervised learning for effective Italian word sense disambiguation, in: Symposium on Semantics in Systems for Text Processing, STEP 2008, Venice, Italy, volume 1.

[3] J. Chen, J. Liu, Combining conceptnet and wordnet for word sense disambiguation, in: IJCNLP 2011.

[4] K. Fragos, I. Maistros, C. Skourlas, Word sense disambiguation using wordnet relations, in: Proceedings of 1st Balkan Conference in Informatics. Thessaloniki, Greece.

[5] multiwordnet, MultiWordNet, http://multiwordnet.itc.it/english/home.php.

[6] C. Musto, F. Narducci, P. Basile, P. Lops, M. de Gemmis, G. Semeraro, Comparing word sense disambiguation and distributional models for cross-language information filtering, in: Proceedings of the 3th Italian Information Retrieval Workshop (IIR 2012), Bari, Italy, pp. 117-120.

[7] E. Pianta, L. Bentivogli, C. Girardi, Multiwordnet: developing an aligned multilingual database, in: Proceedings of the First International Conference on Global WordNet, Mysore, India, pp. 293-302.

[8] D. Reforgiato, A new unsupervised method for document clustering by using wordnet lexical and conceptual relations, in: Journal of Information Retrieval, Springer Netherlands, pp. 563-579.

[9] M. F. Porter, An algorithm for suffix stripping, Program 14 (1980) 130-137.

[10] G. Salton, J. McGill, Introduction to modern information retrieval, in: McGraw-Hill, ISBN 0070544840.

[11] A. Sieminski, Wordnet based word sense disambiguation, in: Proceedings of the Third international conference on Computational collective intelligence: technologies and applications - Volume Part II, ICCCI'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 405-414.

[12] K.-T. Sun, Y.-M. Huang, M.-C. Liu, A wordnet-based near-synonyms and similar-looking word learning system, Proceedings of Educational Technology and Society 14 (2011) 121-134.

[13] WordNet, http://wordnet.princeton.edu/, 1996.