A Linguistic Graph-based Approach for Web News Sentence Searching

Kim Schouten and Flavius Frasincar

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR
Rotterdam, the Netherlands
{schouten,frasincar}@ese.eur.nl

Abstract. With an ever increasing amount of news being published every day, being able to effectively search these vast amounts of information is of primary interest to many Web ventures. As word-based approaches have their limits in that they ignore a lot of the information in texts, we present Destiny, a linguistic approach where news item sentences are represented as a graph featuring disambiguated words as nodes and grammatical relations between words as edges. Searching is then reminiscent of finding an approximate sub-graph isomorphism between the query sentence graph and the graphs representing the news item sentences, exploiting word synonymy, word hypernymy, and sentence grammar. Using a custom corpus of user-rated queries and sentences, the search algorithm is evaluated based on the Mean Average Precision, Spearman's Rho, and the normalized Discounted Cumulative Gain. Compared to the TF-IDF baseline, the Destiny algorithm performs significantly better on these metrics.

1 Introduction

With news information volumes that are already overwhelming, a lot of the human mental activity is nowadays devoted to gathering, filtering, and consuming news information. Ingenious as we humans are, we have developed crude ways of filtering information quickly, using only a fraction of the time actually needed to process all the information accurately. However, reading only the titles or reading a text in a 'quick-and-dirty' fashion only goes so far. While the trade-off between speed and accuracy can probably not be broken due to the limitations of our brain, we do have the possibility to assist ourselves with tools in order to attain results that were previously impossible. One of these possibilities we would like to consider is to search for sentences, thus searching both within documents and across documents.

Despite its simplicity, experience has shown that methods based on TF-IDF [11], or similar metrics where a document is modeled as a bag of words, performs good in many circumstances. In our evaluation we therefore use TF-IDF as a baseline to test our approach against.

To better deal with the peculiarities of language, multiple approaches have been proposed where text is not regarded as a simple collection of words, but where, instead, text is processed using natural language processing techniques to extract valuable information that is implicitly encoded in its structure. A good example is the Hermes News Portal [12], where news items are annotated, linking found lexical representations to instances in an ontology. Queries, comprised of a selected set of concepts from this ontology knowledge base, can then be executed to get the news items pertaining to the entities in the query.

Unfortunately, there is an intrinsic problem with ontology-based approaches. Because not text, but the ontology is used for search, concepts that are not specified in the ontology cannot be found. An ontology thus makes the approach domain dependent. Furthermore, since an ontology is formally specified, information in the text has to be transformed to the logical form of the ontology. This is known to be difficult and sometimes even impossible since there are sentences that cannot be symbolized using first-order logic [1], and most ontologies use propositional logic or description logic (e.g., the many variants of OWL) which provides even less expressive power than first-order logic.

Thus, given that the meaning of language is both infeasible to extract and to represent, one can aim for a processing level just below the semantic interpretation phase [10]. Using the principles of homophonic meaning-representation and compositionality coming from the philosophy of language [3], we can represent language as an interconnected collection of disambiguated words, where the connections represent the grammatical relations that exist between words in a sentence. The interconnected collection of disambiguated words can then naturally be represented by a graph model, with nodes representing the words and edges representing the grammatical relations.

When both the news items and the sentence describing the user query are represented by the above type of graphs, the problem of searching for the query answers in the set of news items becomes related to the sub-graph isomorphism problem. While the standard graph isomorphism problem can be solved by Ullmann's algorithm [13], there are some additional considerations that prevent us from straightway applying it in this situation, the main one being that we are not only interested in full matches, but also in partial matches. We therefore want to measure the degree of similarity between the query sentence graph and all news item sentence graphs, and return a ranked list of sentences in the set of news items that are most similar to the user query.

2 The Destiny Framework

Based on the considerations in the previous section, our task is twofold: first, a process needs to be developed to transform raw text into a graph-based representation using the grammatical relations between words, and second, an algorithm needs to be devised that can compare graphs and compute a similarity score to enable ranking news sentences with respect to a query sentence. The Des-

tiny framework is designed to incorporate both news item processing and query execution on the set of processed news items.

2.1 News Processing

To transform raw text into a dependencies-based graph representation, we have developed a pipeline consisting of various components with their own specific task. The same pipeline is used to process both news items and user queries. In Figure 1, a schematic overview of the framework is given. On the left hand side, the process of news item transformation is shown, while on the right hand side, the process of user query transformation and searching is depicted, each using the same processing pipeline in the middle.

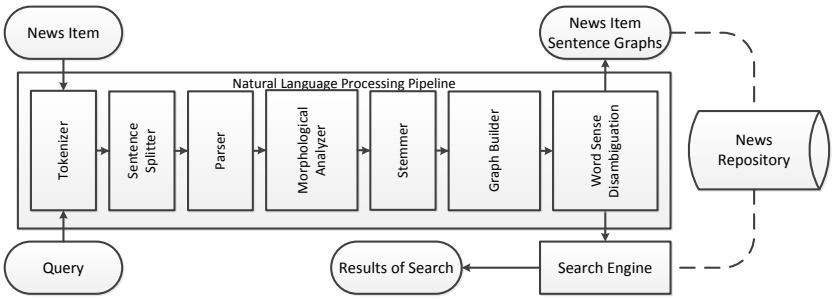

Fig. 1. Conceptual representation of framework

The tokenizer and sentence splitter, both standard components in the GATE framework [2], respectively, determine the word and sentence boundaries, after which the Stanford Parser [8] can extract the dependencies between words as well as the Part-of-Speech (POS) tags. Determining the lemma is then performed by the morphological analyzer, also a standard GATE component, and Porter's stemmer [9] algorithm is used to get the word stems.

Based on the information gathered so far, a graph representation of the sentence is built. To that end, we generate a node for each word, and connect these nodes with directed edges built from the syntactical dependencies output of the parser. Each node holds all information with respect to the word it represents: the literal word, lemma, stem, and POS tag. Each edge is labeled with the type of dependency that exists between these two nodes. As words in a sentence are uniquely represented by a node, the same word appearing multiple times in a sentence will result in multiple nodes, one node for each word.

Since words can have multiple senses, just comparing words based on their lexical representation will result in significant information loss. Therefore, the correct sense of each word has to be determined. This information is used in the search algorithm when determining synonym or hypernym relations between words. As the development of a word sense disambiguation algorithm is not

the core topic of this paper, we have chosen to implement an existing one: the simplified Lesk algorithm [7].

2.2 News Searching

We devise a recursive algorithm for news searching, and the first important issue is to determine the starting point for the algorithm. As sentences can have different structures it is often not possible to compare two sentences by simply starting at the root. However, separately recursing from each node in the graph is not efficient. To combine efficiency with accuracy, only nouns and verbs in the query sentence graph are selected as they are the most probable good fits, the intuition being that these words are most rich in information compared to other types of words. Each one of these nodes in the query graph is used to find similar nodes in the news item graphs using an index on the stem values of all news item nodes. For each matching combination of a node in the query graph and a similar node in the news item graph, the recursion is performed with that combination as its starting point. All scores from the various recursion runs are aggregated on a sentence level so that to each combination of query and news item sentence, only the highest score is associated.

The algorithm compares the two graphs by starting to compare the two nodes in the query graph and a news item graph, respectively. After that, it will compare the nodes they are linked to, recursively traversing the graph until the comparison is complete. Because a node can have multiple parents and multiple children, the algorithm will try to compare edges for both directions. It is however constrained to only compare edges having the same direction.

There are multiple parameters in this algorithm that can be optimized, for example the various features that are used to assign similarity scores to nodes and edges. This optimization has been done using a genetic algorithm which will be explained later in this section. Besides feature weights, another parameter is used to control whether the recursion should continue in a given direction. If there is no direction that has a score (i.e., the similarity score of the edge and the node it connects to, but not any further in the graph) higher than the threshold as given by the parameter, than the algorithm will not recurse any further here.

Both the comparison of nodes and edges contributes to the total similarity score. With features weighted by the genetic optimization, the similarity score of a node or edge is the sum of all matching features. While edges only have one feature, the label denoting the type of grammatical relation, nodes have multiple features.

The five basic features are Boolean comparators on stem, lemma, the full word (i.e., including affixes and suffixes), basic POS category, and detailed POS category. The basic POS category consists of categories like noun, verb, adjective, etc., while the detailed POS category uses inflections like verb tenses and noun number (singular or plural) to create more fine-grained POS categories. These rather simplistic comparators are complemented by a check on synonymy and hypernymy using the senses acquired by the word sense disambiguation component and WordNet. When words are in the same broad POS category (e.g.,

nouns, verbs, adjective, and adverbs), but do not have the same lemma, they are possibly semantically related by synonymy or hypernymy.

For synonymy, WordNet is used to check whether both words with that particular sense are in the same synset, where a synset is used by WordNet to group words with the same meaning. For hypernymy, WordNet is used to find a relation of generalization between the two words. Such a relation does not have to be direct, but can also be indirect, involving multiple words as intermediary steps. For example, 'entity' is the most generalized form of 'car', even though there are multiple words between 'entity' and 'car' that are thus specializations of 'entity' but still generalizations of 'car'. As the strength of the generalization is dependent on the length of the path between the two words, we use the heuristic that the score as determined by the genetic algorithm is divided by the length of the shortest hypernymy path.

The last step in computing the similarity score of a node is an adjustment factor, which is a value between zero and one denoting how regular this word is in the news database. Being the inverse of the number of occurrences of this word in the database, the factor will be zero for the most common word, one for a word which occurs only once, and somewhere in between for the other cases.

2.3 Genetic Optimization

In order to optimize all parameters, a genetic algorithm was employed to attain good parameter settings. In Table 1, an overview is given of all parameters and of the weights assigned to it by the genetic algorithm when applied on the training set. All weights have values in the interval of 0 and 255. The fitness function the genetic algorithm maximizes is the normalized Discounted Cumulative Gain, one of the metrics used in the evaluation. The genetic algorithm itself is a standard implementation, using random initialization, cross-over, mutation, and elitist selection.

Interestingly, there seem to be many local optima for this specific optimization task, as the various cross-validation folds of the genetic algorithm, while roughly resulting in the same performance, yielded very different weights. The many local optima are probably due to the redundancy in natural language. For example, much of the same information can be found in both the literal word, as well as in the stem and the lemma. The same is true of a word and its context. Given the context, one is often able to guess a missing word, which means that much of the information in that word is already covered in the rest of the sentence. Given this redundancy, it is of no surprise that the features available to the genetic algorithm have a non-zero correlation.

In spite of the high standard deviations on the set of weights acquired by the genetic algorithm, there are still some interesting patterns that can be discovered. For example, we can see that the genetic algorithm clearly prefers the basic POS tags over the detailed ones. Also interesting is the fact that the fully inflected word is preferred over the lemma, which in turn is preferred over the stem of the word. While the fully inflected word represents more information than the lemma, just like the lemma represents more information than the stem, using

Table 1. Optimized features and their weights.

feature	one set of weights	**avg** weights	**stdev** weights
search threshold	12.44	95.64	67.42
node: word	182.11	125.30	65.75
node: lemma	50.24	105.11	67.53
node: stem	78.18	82.54	76.20
node: basic POS tag	206.46	119.00	77.96
node: detailed POS tag	2.84	50.62	59.76
node: synonym	106.20	136.90	71.53
node: hypernym	232.88	129.65	64.97
edge: label	200.69	171.82	71.58
significance factor importance	217.32	109.78	70.68

the inflected form of a word makes it less generalizable than using the stem or even the lemma of a word.

3 Evaluation

The results of Destiny are evaluated against a user-defined standard and compared with a classical text-based search algorithm. To that end, we have created a news database, containing 1019 sentences, originating from 19 news items, and 10 query sentences. Queries are constructed by rewriting sentences from the set of news item sentences. In rewriting, the meaning of the original sentence was kept the same as much as possible, but both words and word order were changed (for example by introducing synonyms and swapping the subject-object order). Each combination of a news sentence and a query was rated by at least three different persons. This led to a data set of over 30,500 data points (10 queries $\times$ 1019 sentences $\times$ at least 3 scores).

The various parameters of the search engine were optimized using a genetic algorithm, as discussed in the previous section. Therefore, the data set was split into a training and a test set. Because the results for each query are rather limited, the data set was split on the query level: 5 queries and their results went into the training set and the other 5 queries comprised the test set. To allow for cross-validation, 32 different splits were made. Because some queries have more results than others, all splits were checked to be balanced in the number of query results. In this way, the quantity of search results could not influence the quality of the results.

The baseline to compare our results against is the standard TF-IDF algorithm. All TF-IDF sores are computed over the test set for each of the 32 splits. As TF-IDF does not have to be trained, the training set was not used. Three metrics are used to compare the Destiny algorithm with the TF-IDF baseline,

which are the Mean Average Precision (MAP), Spearman's Rho, and the normalized Discounted Cumulative Gain (nDCG) [6].

The main concern when using MAP is that it is Boolean with respect to the relevance of a certain sentence given a query, while the user scores return a gradation of relevance. This means that in order to compute the MAP, the user scores have to split into two groups: relevant and not relevant, based on some cut-off relevance score c. Since this value is arbitrary, we have chosen to compute an average MAP score over all possible values of c, from 0 to 3 with a step size of 0.1.

The results, shown in Table 2, clearly show that Destiny significantly outperforms the TF-IDF baseline. For nDCG and Spearman's Rho, the p-value is computed for the paired one-sided t-test on the two sets of scores consisting of the 32 split scores for both Destiny and TF-IDF, respectively. For MAP, because we computed the average over all cut-off values, the same t-test is computed over 30 cut-off values $\times$ 32 folds which results in 960 split scores.

Table 2. Evaluation results

	TF-IDF mean score	Destiny mean score	rel. improvement	t-test p-value
nDCG	0.238	0.253	11.2%	< 0.001
MAP	0.376	0.424	12.8%	< 0.001
Sp. Rho	0.215	0.282	31.6%	< 0.001

4 Concluding Remarks

By developing and implementing Destiny, we have shown that it is feasible to search news sentences in a linguistic fashion. Using a natural language processing pipeline, both news items and queries are transformed into a graph representation, which are then compared to each other using the degree of sub-graph isomorphism as a measure for similarity. By representing text as a graph, the original semantic relatedness between words in the sentence is preserved, thus allowing the search engine to utilize this information.

Words, represented as nodes in the graph, are compared not only lexically, but also semantically by means of a word sense disambiguation algorithm present in the natural language processing pipeline. This allows for checks on both synonymy and hypernymy between words. The Mean Average Precision, Spearman's Rho, and normalized Discounted Cumulative Gain achieved by Destiny are significantly better than the scores obtained from the TF-IDF baseline.

As future work we would like to improve the accuracy of the search results by adding named entity recognition and co-reference resolution. The graph-based approach from [5] seems especially suitable for co-reference resolution. Furthermore, we would like to investigate the benefits of using a graph edit distance [4]

measure, to mitigate problems with varying graph structures which are now not processed correctly.

Acknowledgment

The authors are partially supported by the Dutch national program COMMIT.

References

1. J. Barwise and R. Cooper. Generalized Quantifiers and Natural Language. *Linguistics and Philosophy*, 4:159–219, 1981.
2. H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. University of Sheffield Department of Computer Science, 2011.
3. M. Devitt and R. Hanley, editors. *The Blackwell Guide to the Philosophy of Language*. Blackwell Publishing, 2006.
4. R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *7th International Conference on Business Process Management (BPM 2009)*, volume 5701, pages 48–63. Springer, 2009.
5. A. Haghighi and D. Klein. Coreference Resolution in a Modular, Entity-Centered Model. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2010)*, pages 385–393. ACL, 2010.
6. K. Järvelin and J. Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.
7. A. Kilgarriff and J. Rosenzweig. English SENSEVAL: Report and Results. In *2nd International Conference on Language Resources and Evaluation (LREC 2000)*, pages 1239–1244. ELRA, 2000.
8. D. Klein and C. Manning. Accurate Unlexicalized Parsing. In *41st Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 423–430. ACL, 2003.
9. M. F. Porter. An Algorithm for Suffix Stripping. In *Readings in Information Retrieval*, pages 313–316. Morgan Kaufmann Publishers Inc., 1997.
10. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.
11. G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
12. K. Schouten, P. Ruijgrok, J. Borsje, F. Frasincar, L. Levering, and F. Hogenboom. A Semantic Web-based Approach for Personalizing News. In *ACM Symposium on Applied Computing (SAC 2010)*, pages 854–861. ACM, 2010.
13. J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *J. ACM*, 23(1):31–42, 1976.