View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 25 (2016) 272 – 279

Procedia Technology

Global Colloquium in Recent Advancement and Effectual Researches in Engineering, Science and Technology (RAEREST 2016)

Evaluation of Scalable Database Driven Reverse Dictionary

Soumya Rajan[a*], Kumary R Soumya[a]

[a]*Department of Computer Science & Engineering, Jyothi Engineering College, Cheruthuruthy, Thrissur, India*

Abstract

While a traditional forward dictionary maps words to their definitions, a reverse dictionary takes a user input phrase with a desired concept, and returns a set of candidate words closely related to the input phrase. This application is significant not only for the general public, but mainly to those who work personally with words. It is also important in the general field of conceptual search. Upon receiving a search concept, the Reverse Dictionary consults the forward dictionary and selects those words whose definitions are similar to the given concept. And thus it is reduced to a concept similarity problem. In this paper, different concept similarity measures are compared and the best among them is proposed. The experimental results shows that the approach used here provides significant improvements in performance level without losing the quality of the result.

© 2016 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the organizing committee of RAEREST 2016

Keywords: Dictionaries; Stemming; Stop Words; Pos Tagging; Concept Mining,

1. Introduction and Related Works

When a regular (forward) dictionary maps words to their definitions, a Reverse Dictionary(RD) performs the reverse mapping. That is, given a phrase describing a desired concept, a Reverse Dictionary provides words whose definitions match the entered definition phrase. Thus for an example, when a forward dictionary returns the meaning of the word "sorrow" as "sadness", a reverse dictionary recommends the user to enter the phrase "a feeling of deep grief" or "a sense of deep pain" as input, and expects to receive the word "sorrow" and probably other conceptually similar words as output.

We usually have words on the tip of our tongue, but we can't quite remember it. And that is where the problem lies. The category of people mainly affected by this problem is writers, including professional writers, students, teachers, researchers etc. For most people with a certain level of education, the problem is not the lack of knowledge about the meaning of a word, but, being unable to recall the appropriate word at the time of requirement.

2212-0173 © 2016 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the organizing committee of RAEREST 2016
doi:10.1016/j.protcy.2016.08.107

The RD solves this widespread problem[1].

1.1. RD Problem Approach

In RD, upon receiving a search concept, it consults the forward dictionary and selects those words whose definitions are similar to the searched concept. These words then form the output of the RD lookup. The problem then reduces to a concept similarity problem (CSP). Particularly in computer science, concept similarity has been dealt by both Information Retrieval (IR) researchers [2], [3] as well as Database researchers [4]. The main hitch here is that the input of the user is not likely to exactly match the definition of a word in the forward dictionary and the response efficiency needs to be similar to that of forward dictionary online lookups. According to a recent Forrester study, end users become impatient if a website takes longer than 4-5 seconds to respond to a request [5].

Most of the studies regarding the similarity of concepts, model concepts as single words. For example works in text classification, examined intensively in [6], cluster similar documents if they contain co-occurring words . It doesn't consider sentences or phrases as such. In existing word sense disambiguation approaches, researchers search for the contextual meaning of a polysemous word (i.e., a word with many meanings) based on nearby words in the sentence where the target word appears. In such case too it considers a single word at a time. This single word emphasis is well identified in the literature [3], [7]. Where as in RD, semantic similarities must be computed between multiword phrases. And this is well addressed in paper[8].

1.2. Concept Similarity Problem

The semantic similarity between concepts is a measure of semantic distance between two concepts which can be estimated by using ontologies. Semantic similarity can be used to identify concepts with common "characteristics". Even though it is difficult to give a formal definition for relatedness between concepts, we can find the relatedness between them. For example, a small child can say that “apple” and “grapes” are more related to each other than “apple” and “tomatoes”. And these pairs of concepts are related to each other and its structure definition is formally called “is-a” hierarchy[9]. Here we examine three semantic similarity methods and choose one among them.

1.2.1. Hirst and St-Onge Measure (HSO)

HSO[10] measure calculates similarity between concepts using the path distance between the concept nodes in the taxonomy, number of changes in direction of the path connecting two concepts and the allowableness of the path. If there is a close relation between meanings of two concepts or words, then the concepts are said to be semantically related to each other. An Allowable Path is a path that does not deviate away from the meaning of the source concept and thus is considered in the calculation of relatedness. Let ‘d’ be the number of changes of direction in the path that relates two concepts C1 and C2, and C, k are constants whose values are derived through experiments. And ‘len’ is the short path relating (i.e minimum number of links) concepts C1 to concept C2.Then similarity function of HSO is formulated as follows:

$$Sim_{HSO}(C1,C2) = C\text{-}len(C1,C2) - k*d \quad (1)$$

In brief according HSO, two lexicalized concepts are semantically close if their WordNet synsets are connected by a path which is not too long and which does not change direction too often.

1.2.2. Wu and Palmer(wup)

In Wu and Palmer[11], let C1 and C2 be two concepts in the taxonomy, this similarity measure considers the position of C1 and C2 to the position of the least common sub-sumer(LCS) C, that is the most specific common concept. Several parents can be shared by C1 and C2 by multiple paths. The most specific common concept is the closest common ancestor C (the common parent related with the minimum number of IS-A links with concepts C1 and C2).

$$Sim_{wup}(C1,C2) = (2*N)/(N1+N2+2*N) \quad (2)$$

Where N1 and N2 are the distance (number of IS-A links) that separates the concept C1 and C2 from LCS and N is the distance which separates the LCS of C1 and C2 from the root node

1.2.3. Leacock and chodorow(LC)

The relatedness similarity measure proposed by Leacock and Chodorow(LC)[12] is

$$Sim_{LC}(C1,C2) = -log(length/(2*D)) \quad (3)$$

Where *length* is the length of the shortest path between the two concepts (using node-counting) and D is the maximum depth of the taxonomy. Based on this measure, the shortest path between two concepts of the ontology restrict to taxonomic links is normalized by introducing a division by the double of the maximum hierarchy depth.

2. Solution Approach

In RD, users might input a phrase describing an unknown term of interest. RD should return a set of possible matches from which the users select their choice of terms. This is complex, however, because the user is unlikely to enter a definition that exactly matches one found in a dictionary.

Firstly we define several concepts that will be useful in describing the method used[1]. Term t is any valid word in the English language, e.g., "cow," "boy," "reading," and "merrily". Phrase P is sequence of one or more terms. P is given by P = <t1, t2, ….., ti,….tn> where t1,t2…. are elements of P.

A dictionary D is a set of mappings from phrase to phrase. It is a mapping from word phrases (W) to sense phrases (S), i.e W →S. It denotes sense phrase is the meaning of the word phrase. For example: "indispensable" → "absolutely necessary." This denotes the fact that the meaning of the phrase "indispensable" means the phrase "absolutely necessary."

Forward mapping: a forward mapping assigns all the senses for a particular word phrase. This is expressed in terms of a forward map set (FMS). The FMS of a (word) phrase W, denoted by F(W) is the set of (sense) phrases {S1,S2….Sx} such that for each Sj ε F(Wi), (Wi → Sj) ε D. For example, the term "blue" is associated with various meanings, including "a color or pigment" and "melancholy or depressed." Here, F(blue) will contain both of these phrases.

Reverse mapping: Reverse mapping is applied to the terms and is represented as a reverse map set (RMS). The reverse map set of a term t consists of all the words in whose definition t appears. For example, the word "interest" appear in the definitions of both "enthusiasm," which can be defined as "a lively interest" and "apathetic," which can be defined as "marked by a lack of interest." Here, R(interest) would include both "enthusiasm" and "apathetic."

The words "manage" and "managed" are conceptually equivalent. For concept matching we apply standard stemming algorithms which reduces each word to its base form.

Since, a user is unlikely to enter an exact dictionary definition, but may enter something conceptually similar to the dictionary meaning, we need describe how words can be conceptually related to one another. And for that we define several types of relatedness below.

Synonym set: A set of conceptually related terms for t. It is represented as $W_{syn}(t)=\{t_1,t_2,\ldots\ldots t_j,\ldots..,t_n\}$. An example is W_{syn}(Beautiful)={ Pretty, Attractive, Lovely, Stunning}.

Antonym set: A set of conceptually opposite or negated terms for t. It is represented as $W_{ant}(t) = \{t_1,t_2,\ldots\ldots t_j,\ldots..,t_n\}$. An example is W_{ant}(appear) ={" disappear", " vanish"}.

Hypernym set: A set of conceptually more general terms describing t. It is represented as $W_{hyr}(t)= \{t_1,t_2,\ldots\ldots t_j,\ldots..,t_n\}$. An example is W_{hyr}(apple)={"fruit"} .

Hyponym Set: A set of conceptually more specific terms describing t. It is represented as $W_{hyo}(t)=\{t_1,t_2,\ldots\ldots t_j,\ldots..,t_n\}$, An example is W_{hyo}(animal)={"dog", "horse", "elephant"}.

Forward mapping sets, as well as synonym, antonym, hypernym, and hyponym sets can be retrieved from an existing corpus. In this paper, we draw these sets from the WordNet [13] database.

User phrase: A sequence of terms input by a user is referred as user phrase U.

Stop word sets: Some common words which are not useful for the purposes of indexing are called stop words.

Level 1 stop words (L1), which are always removed during index building and querying.

Level 2 stop words (L2), which may or may not be useful in indexing. An example of an area where a word may or may not be useful is gender, because the gender is important in some cases but not others. Eg: "man who is married" should logically return the word "husband," but not "wife;"

Negation word set: a user might enter an input phrase using the word "not," in such case antonym of the negated term would be more accurate.

Table 1. Some example of stop words.

Level 1 stop words	Level 2 stop words	Negation word
a, an, are, as, at, by, is, it, its, on, was, were, will, be, person, some, someone, too, very, who, the ,in, of, and, to, that, for, with, this, from, which, when, what, than, into, these, where, those, how, during, without, upon, toward, among, beside, whose, whom, onto, anybody, whenever, whereas	Women, female, male	Seldom, no, none, not, nor, without, hardly, never, lack, lacking, nowhere

Query: A query phrase Q is a Boolean expression based on an input U.

Output: The output O of this method consists of a set of Ws, such that a definition of each W in the output satisfies Q.

2.1. Solution Overview

The reverse dictionary application takes the sense phrase as input and partition the sense phrase into separate words. Various steps involved in the implementation of reverse dictionary includes lexical analysis, elimination of stop words, stemming, pos tagging, concept mining, building the reverse mapping set, querying the reverse mapping set and ranking candidate words[1]. The stop words are removed to increase the overall scalability of the system. The core terms of the input phrase is obtained by running each word through a standard stemming algorithm e.g., the Porter stemmer [14]. Pos tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context. Concept mining is associated with understanding the meaning of words or phrases. Building the reverse mapping set is a single time event and this is well addressed below. Querying the reverse mapping set results in the generation of candidate words. We need to sort these words based on the similarity to the user concept. For similarity Calculation, we need to calculate term similarity and Syntactical similarity. For that we need to assign a similarity measure for each sense and user input phrase. We use Word Net dictionary to estimate the similarity score. It can be used as a lexical ontology for natural language terms.

The find candidate words phase comprises of two key sub steps[1] such as building the RMS; and querying RMS. The RMS is build once for the entire words in the dictionary, because every time creating the RMS is not a feasible solution. While querying the RMS candidate words are obtained that are conceptually similar to the input phrase given. These words are then sorted based on a similarity measure. And this is the output of the system.

The generation of candidate words phase consists of two key substeps: 1) build the RMS; and 2) query the RMS. The reverse mapping set (RMS) of a term t, denoted R(t), is a set of phrases { P1, P2, Pi,……, Pm}, such that Pi ϵ R(t), t ϵ F(Pi). So, the reverse map set of a term t consists of all the (word) phrases in whose definition t appears.

2.2.Building Reverse Mapping Sets

Building the RMS of a term *t*, R{t}, is a matter of finding all Ws in whose definition *t* appears. For large size of dictionaries, creating such mappings on the fly is infeasible. Thus we do reverse mapping for every relevant term in the dictionary. This is a one time, offline event. Once these mappings exist, we can use them for ongoing lookup. Thus creating the corpus has no effect on runtime performance.

INPUT : A dictionary D

OUTPUT : Reverse mappings for all terms appearing in the sense phases(definitions) in D

1. Get the definition phrase from Dictionary D.
2. Extract Sentences from Sentence Boundary Detection using vanilla approach.
3. $\{s_1, s_2, s_3...\}$ is set of sentence appears in the definition of P_i
4. Apply POS Tagging for Sentence S_i.
5. Stop word removal function on S_i
6. Apply Stemming and extract terms from S_i.
7. Find Synonyms and hypernyms of each term from dictionary.
8. Add related term set to extracted terms.

2.3.Querying The Reverse Mapping Sets

Upon receiving input phrase, we query the reverse indexes already present in the database to find candidate words whose definitions have any similarity to the input phrase. For a input phrase “mind blowing” first extract the core terms present in this phrase: “mind” and “blowing”. Then consult the appropriate R indexes, R(mind) and R(blowing) and find those words in whose definitions the words “mind” and “blowing” occur simultaneously. Each such word becomes a candidate word.

If this first step does not generate a sufficient number of output Ws we then broaden the scope of query Q to include the synonyms, hyponyms, and hypernyms of the terms in Q. If these steps still do not generate a sufficient number of output Ws, we remove terms from Q to increase the scope of possible outputs until a sufficient no: of output Ws has been generated. Then sort the results based on similarity to U. Return the top required no: of Ws. We normalize input phrase by extracting necessary terms using NLP based word extractor .We also apply stemming on the terms and also avoid negated term by its antonym.

INPUT : U (user input phrase)
alpha**,** a tuneable input parameter, which represents the minimum number of Word phrases needed to halt processing and return output.

OUTPUT : Candidate word set

1. Extract terms from U
2. $T_s= \{t_1,t_2..t_i\}$ appears in User Input.
   if negation word is present, replace negated term t by its antonym(t).
3. Find w={w1,w2…wi}, find all r(t) from each term in Term Set Ts, candidate words
4. If this step does not generate a sufficient number of output W_s, defined by alpa
   Expand related terms (r_{ti}) from term set t_i.
5. Generate word set from r_{ti}, If sizeOf(word set)>alpha

2.4. Ranking Candidate Words

Here semantic similarity of definition of candidate words found is compared with the user input phrase U. On the basis of that, sorts a set of output words in the order of decreasing similarity to U as compared to the candidate words.

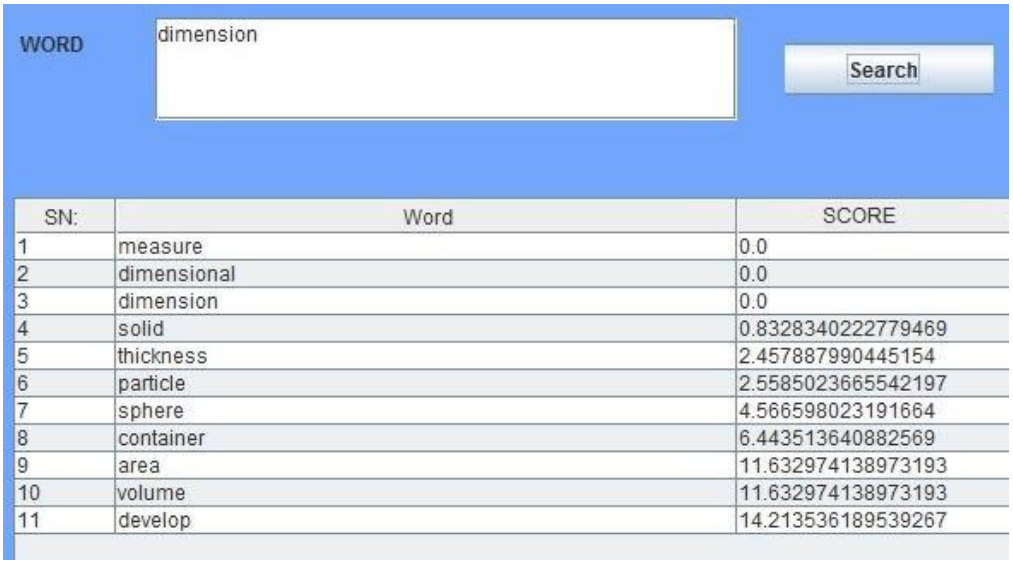
INPUT : User Input Terms, Candidate word set Ws $[w_1,w_2..w_n]$, $Ts= \{t_1,t_2..t_i\}$ appears in User Input

OUTPUT : Sorted word set.

1. For(n =0 to numberOfwords(Ws))
2. Calculate Term similarity Of Wn
   WT_s=Mapped Terms of W_n, TW_n Union T_s
   Term Similarity= Sum (LC (W_n,WT_{si})) , where WTsi=[t_1,t_2,..t_n].
   Where LC is shortest path calculated between two synsets for their measure of similarity.
3. Sort word set W_s based on term similarity.

3. Result

On giving a search concept we get conceptually similar outputs in the decreasing order of its similarity with the search concept provided.



SN:	Word	SCORE
1	measure	0.0
2	dimensional	0.0
3	dimension	0.0
4	solid	0.8328340222779469
5	thickness	2.457887990445154
6	particle	2.5585023665542197
7	sphere	4.566598023191664
8	container	6.443513640882569
9	area	11.632974138973193
10	volume	11.632974138973193
11	develop	14.213536189539267

Fig. 1. Shows the Result values for a given sample.

3.1. Performance Overview

Here the results of experiments are reported to judge solution quality. To determine the optimal result sets for 100 experimental query strings, the optimal results are computed manually as a benchmark for the accuracy of our algorithms. We also have written methods in our system to retrieve the response time for all the three types of similarity measure discussed above. And it was found that Leacock and Chodorow similarity measure constituted the least average response time and that is shown in Fig.2. The solution quality is measured with the standard Information Retrieval metrics precision and recall.

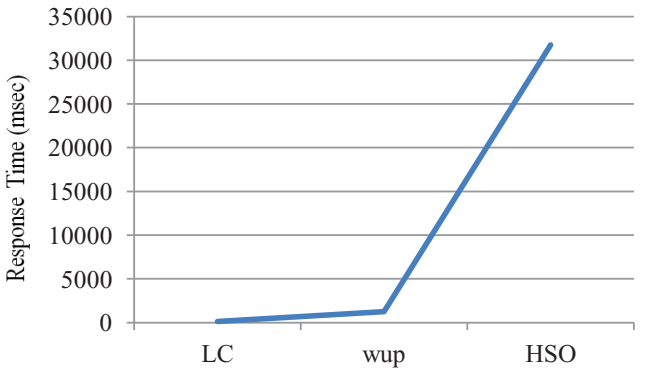


Fig. 2. Shows that average response time of LC is the least on a sample of 100 words.

Precision =No: of similar words retrieved / Total no:of of words retrieved.
Recall= No: of similar words retrieved / No: of similar words in the collection.
The Precision and Recall values for this reverse dictionary on the three concept similarity measure gives the following results in graph.

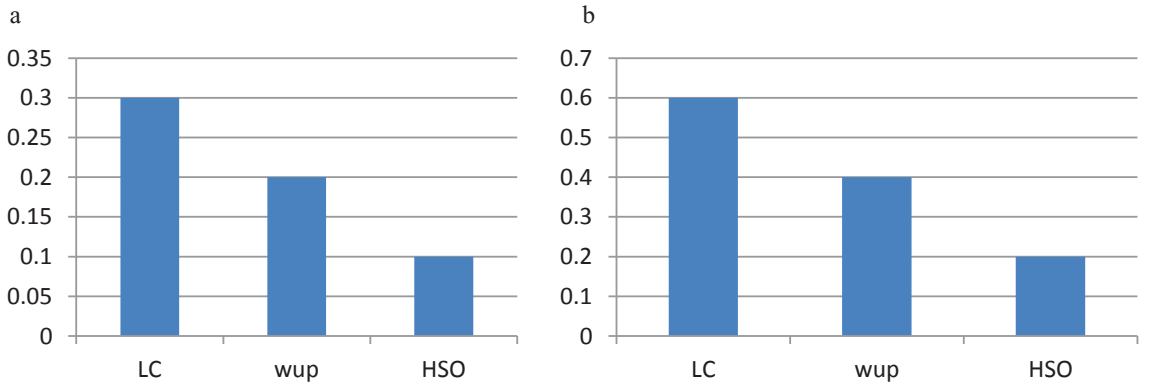


Fig. 3. Shows the Precision(a) and Recall (b)values for this reverse dictionary application on the three concept similarity measure discussed above.

3. Conclusion

The significant challenges inherent in building a reverse dictionary is studied here. The problem is then mapped to the well-known conceptual similarity problem. Here discussed a set of methods for building and querying a reverse dictionary. And the performance evaluation of this shows that it performs well, even under load without sacrificing solution quality. We can implement this system for large scale data by applying map reducing algorithms.

2. 3. Future Works

Existing reverse dictionary approaches typically split a sentence into a word sequence with the help of syntactic chunker, which does not effectively handle the inconsistent meaning between a phrase and the words it contains,

such as {“come to the point “,“a great deal of,” “great”}. We address this issue by developing a learning approach for candidate word generation in reverse dictionary .

A typical kind of error occuring in our current approach is caused by the inconsistent meaning combination between a phrase and the words it contains, such as {“not bad,” “bad”} and {“a great deal of,” “great”}. The bag-of-words and syntactic chunkers are not effective enough to handle this inconsistency phenomenon. For example, bag-of-words segmentation regards each word as a separate unit, which does not capture the exact meaning.

So, we can use a joint framework for reverse dictionary, which simultaneously produces useful segmentations and predicts candidate based on the segmentation results. A candidate generation model can be developed to generate the segmentation candidates of a sentence, a segmentation ranking model to score each segmentation result of a given sentence and a classification model to predict each segments. We train the joint framework directly from sentences with meaning words.

References

[1] Anindya Datta, Ryan Shaw, Debra VanderMeer and Kaushik Dutta (2013) ‘Building a Scalable Database-Driven Reverse Dictionary’- VOL. 25, NO.3, pp.528-540

[2] D. Lin, “An Information-Theoretic Definition of Similarity,” Proc.Int’l Conf. Machine Learning, 1998.

[3] R. Mihalcea, C. Corley, and C. Strapparava, “Corpus-Based and Knowledge-Based Measures of Text Semantic Similarity,” Proc. Nat’l Conf. Artificial Intelligence, 2006.

[4] S. Berchtold, D.A. Keim, and H.-P. Kriegel, “Using Extended Feature Objects for Partial Similarity Retrieval,” The VLDB J., vol. 6, no. 4, pp. 333-348, Nov. 1997.

[5] Forrester Consulting, “Ecommerce Web Site Performance Today,” http://www.akamai.com/2seconds, Aug. 2009.

[6] F. Sebastiani, “Machine Learning in Automated Text Categorization,” ACM Computing Surveys, vol. 34, no. 1, pp. 1-47, Mar. 2002.

[7] J. Kim and K. Candan, “Cp/cv: Concept Similarity Mining without Frequency Information from Domain Describing Taxonomies,” Proc. ACM Conf. Information and Knowledge Management, 2006.

[8] T. Dao and T. Simpson, “Measuring Similarity between Sentences,” http://opensvn.csie.org/WordNetDotNet/trunk/Projects/Thanh/Paper/WordNetDotNet_Semantic_Similarity.pdf, 2009.

[9] Thabet Slimani. " Description and Evaluation of Semantic Similarity Measures Approaches", In International Journal of Computer Applications (0975 – 8887), Volume 80 – No.10, October 2013

[10] Hirst G. and St-Onge, D. " Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms", In Proceedings of Fellbaum, pages 305-332,1998.

[11] Z. Wu and M. Palmer, “Verbs Semantics and Lexical Selection,” Proc. 32nd Ann. Meeting Assoc. for Computational Linguistics, pp.133-138, 1994.

[12] Claudia Leacock and Martin Chodorow, "Combining Local Context and WordNet Similarity for Word Sense Identification", chapter 11, pages 265–283,1998

[13] G. Miller, C. Fellbaum, R. Tengi, P. Wakefield, and H. Langone, “Wordnet LexicalDatabase,” http://wordnet.princeton.edu/wordnet/download/, 2009.

[14] M. Porter, “The Porter Stemming Algorithm,” ttp://tartarus.org/martin/ Porter Stemmer/, 2009.