On Tag Spell Checking

Franco Maria Nardini[2], Fabrizio Silvestri[2],
Hossein Vahabi[1,2], Pedram Vahabi[4], and Ophir Frieder[3]

[1] IMT, Lucca, Italy
[2] ISTI-CNR, Pisa, Italy
[3] Department of Computer Science
Georgetown University, Washington DC, USA
[4] University of Modena and Reggio Emilia, Modena, Italy

Abstract. Exploiting the cumulative behavior of users is a common technique used to improve many popular online services. We build a tag spell checker using a graph-based model. In particular, we present a novel technique based on the graph of tags associated with objects made available by online sites such as Flickr and YouTube. We show the effectiveness of our approach on the basis of an experimentation done on real-world data. We show a precision of up to 93% with a recall (i.e., the number of errors detected) of up to 100%.

1 Introduction

Differing from query spell checking, the goal of tag spelling correction is to enable the tagged object to be *actually* retrieved. Correcting "hip hop" as "hip-hop", when the latter is more frequent than the former, is a good way to allow people to find the resource when querying for the concept "hip-hop"[1]. By tagging a resource, a user wants that resource to be easily found. When querying, a user formulates a sentence-like text to retrieve the desired concept and to satisfy her/his information need. On the other hand, with tags, users leave "*breadcrumbs*" for others to detect. Like "*breadcrumbs*", tags do not have any particular inter-relationship apart from the fact that they were left by the same user.

We exploit the collective knowledge [1,2] of users to build a spell checking system on tags. The main challenge is to enable tag spell checkers to manage *sets of terms* (with their relative co-occurrence patterns) instead of strings of terms, namely, queries.

Much previous work is devoted to query spell checking. Differing from queries, namely short strings made up of two or three terms, tags are sets of about ten terms per resource. We exploit this relatively high number of tags per resource to provide correct spelling for tags. Indeed, our method exploits correlation between tags associated with the same resource. We are able to detect and correct

[1] The hidden assumption we do is that people formulate queries for resources, following the same mental process as people tagging resources.

E. Chavez and S. Lonardi (Eds.): SPIRE 2010, LNCS 6393, pp. 37–42, 2010.

common variations of tags by proposing the "right", i.e., the most commonly used, versions.

We evaluate our method through a user study on a set of tagged resource coming from YouTube. Note that our experiments are fully reproducible. Instead of using proprietary data sources, like search engines' query logs, we leverage publicly available resources.

Research on spell checking has focused either on non-word errors or on real-word errors [3]. Non-word errors such as *ohuse* for *house* can easily be detected by validating each word against a lexicon, while real-world errors, e.g., *out* in *I am going our tonight*, are difficult to detect. Cucerzan *et al.* [4] investigate the use of implicit and explicit information about language contained in query logs for spelling correction of search queries. Zhang *et al.* [5] propose an approach to spelling correction based on Aspell. Shaback *et al.* [6] propose a multi-level feature-based framework for spelling correction via machine learning. Merhav *et al.* [7] use a probabilistic approach to enrich descriptors with corrected terms in a P2P application. Ahmad *et al.* [8] apply the noisy channel model to search query spelling corrections.

Unlike previous approaches, we cannot rely on information about sequences of terms, or n-grams. We must, thus, find another way to contextualize tags within other tags that are used in association with the same resource.

2 Model Description

In tagging objects, users associate a set of words, i.e., tags, with a resource, e.g., a video, a photo, or a document, having in mind a precise semantic concept. Actually, tagging is the way users allow their resources to be found. Based on this hypothesis, our spell checker and corrector presents two important features: i) it is able to identify a misspelled tag, ii) it proposes a ranked list of "*right*", i.e., most likely to come to users' minds, tags associated with the misspelled tag.

We use a weighed co-occurrence graph model to capture relationships among tags. Such relationships are exploited to detect a misspelled tag and to identify a list of possibly correct tags.

Let R be a set of resources. Let Σ be a finite alphabet. Let $T \subseteq \Sigma^*$ be a set of tags associated with each resource. Let $\gamma : R \rightarrow T$ be a function from resources to set of tags mapping a resource with its associated set of tags. Furthermore, let $T^* = \cup \{\gamma(r), \forall r \in R\}$ be the union of all tags for all resources in R.

Let $G = (V, E)$ be an undirected graph. V is the set of nodes where each node represents a tag $t \in T^*$, and E is the set of edges defined as $E = V \times V$. Given two nodes, u, v, they share an edge if they are associated at least once with the same resource. More formally, $E = \{(u, v)|u, v \in V$, and $\exists r \in R|u, v \in \gamma(r)\}$. Both edges and nodes in the graph are weighted. Let $u, v \in V$ be two tags. Let $w_e : E \rightarrow \mathbb{R}$ be a weighting function for edges measuring the co-occurrence of the two tags, namely, the number of times the two tags appear together for a resource. For a given node $v \in V$, $w_v : V \rightarrow \mathbb{R}$ associates a tag with its weight.

Given two nodes $u, v \in V$, let $P_{u,v}$ be the set of all paths of any length between u and v. Let $\sigma : V \times V \rightarrow \mathbb{R}$, where $\sigma(u, v) = min_{pathlength} P_{u,v}$ be the function providing the length of the shortest path between the two nodes u, v.

Given a tag $t \in V$, and a threshold value for shortest path l, we define "neighbor nodes" the set of nodes $N_t^l = \{t_1 \in V | \sigma(t_1, t) \leq l\}$. "Neighbor nodes" are then filtered by using the tag frequency. For each node in N_t^l, we select from the set nodes having a frequency greater than the frequency of the tag t.

Let $NG_t^l = \{t_1 \in N_t^l | w_v(t_1) > w_v(t)\}$ be the set of neighbors of t at maximum distance l having a frequency greater than the tag t. Given two nodes u, v, let $d(u, v)$ be a function returning the edit distance of the two tags u, v. By applying d to a tag t and tags in its neighborhood, NG_t^l, we define the "candidate neighbor nodes" as follows.

Definition 1. *Given a tag $t \in V$, and a threshold value for the edit distance δ, we define $F_t = \{t_1 \in NG_t^l | d(t_1, t) \leq \delta\}$ as the "*candidate neighbor node*" set.*

We use the *candidate neighbor node* set to check if the tag t is misspelled and, if needed, to find better tags to be used instead of t. We assume that, if F_t is empty then t is a right tag, and it does not need any correction. This approach allows us to have high effectiveness due to relationships between neighbor nodes. The method is also very efficient as we explore only a part of the graph in to find candidate neighbor nodes.

Our approach to the spelling correction problem using tags is defined as:

> TagSpellingCorrection: Given $s \in \Sigma^*$, find $s' \in T^*$ such that $d(s, s') \leq \delta$ and $P(s'|s) = max_{t \in T^* : d(s,t) \leq \delta} R(t|s)$, where, $d(s, t)$ is a distance function between the two tags, δ is a threshold value, $P(s'|s)$ is the probability of having s' as a correction of s, and $R(t|s) = 1$ if $t \in F_s$, or 0, otherwise.

Algorithm (1) solves the TagSpellingCorrection problem providing a list of possible right tags for a given tag t. It filters the co-occurrence graph by sorting "neighbor nodes" by their importance, and by considering only the top-r most frequent ones.

Given a node $t \in V$, and a $r \in \mathbb{N}$, we define a function $T_p : V \times \mathbb{N} \rightarrow O$ taking the top-r most important nodes of a node t, where $O = \{v_i \in N_t^1, i = 1, ..., r | w_e(v_j) \geq w_e(v_{j+1})$, and $w_e(v_1) = max_{t \in N_t^l}(w_e(t))$ with $j \in 1, ..., r - 1\}$.

3 Experiments

To evaluate our spelling correction approach we built a dataset of tags from YouTube. In particular, we crawled 568,458 distinct YouTube videos obtaining a total of 5,817,896 tags.

Precision and recall are evaluated by means of a *user-study* to get the percentage of good corrections. We asked assessors to evaluate four complete vocabularies produced by four different runs of the Algorithm (1). The four runs differed from the set of parameters used to produce the vocabulary.

Algorithm 1. FindCorrectTag

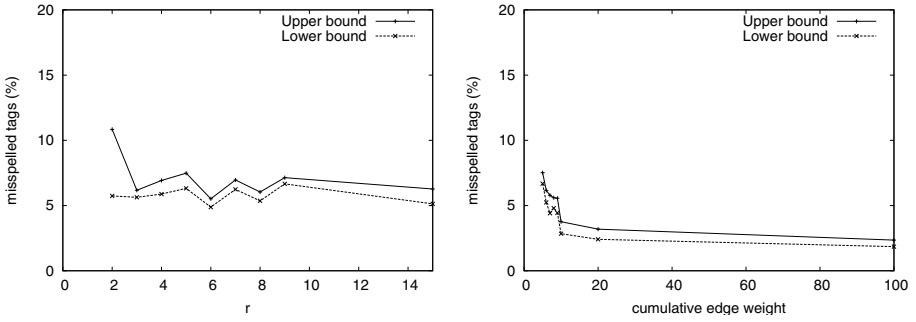
Input: $G = (V, E)$ co-occurrence graph, a tag $t \in V$, a threshold level for shortest path $l > 0$, and a threshold value for edit distance $\delta > 0$, the number of top nodes to consider $r \in \mathbb{N}$, node and edge threshold weights f, k.
Output: a list F_t of correct tags for t.
$F_t = \{\}$, $Temp_s = \{\}$, $s = t$, $V_{f,k} = \{\}$, $E_{f,k} = \{\}$
for all $t \in V$ **do**
  if $(w_v(t) > k)$ **then**
    $V_{f,k} = V_{f,k} \cup \{t\}$
  end if
end for
for all $e \in E$ **do**
  if $(w_e(e) > f)$ **then**
    $E_{f,k} = E_{f,k} \cup \{e\}$
  end if
end for
$LevelwiseBreadthFirst(G_{f,k}, t, l, 1)$;
for all $t_1 \in V_{f,k}$ in $Temp_s$ **do**
  if $(w_v(t_1) > w_v(t)) \wedge (d(t_1, t) > \delta)$ **then**
    $F_t = F_t \cup \{t_1\}$
  end if
end for

To compute recall we performed an estimation of the number of total wrong tags in the collection. We avoided performing a user-study over all the collection of tags by taking samples of dimension $n = 101$ and by computing their average values. The sample was chosen with the Mersenne-Twister algorithm. We computed the *confidence interval* [9] with $\alpha = 0.05$, and the *tstudent* values equal to 1.984.

Figure 1(a) shows the percentage of detected misspellings by varying the top-r most weighted edges of each node. Generally, it decreases as the threshold on the tag frequency increases. Figure 1(b) shows the percentage of estimated misspelled tags by varying the cumulative edge weight. It decreases as the threshold on edges

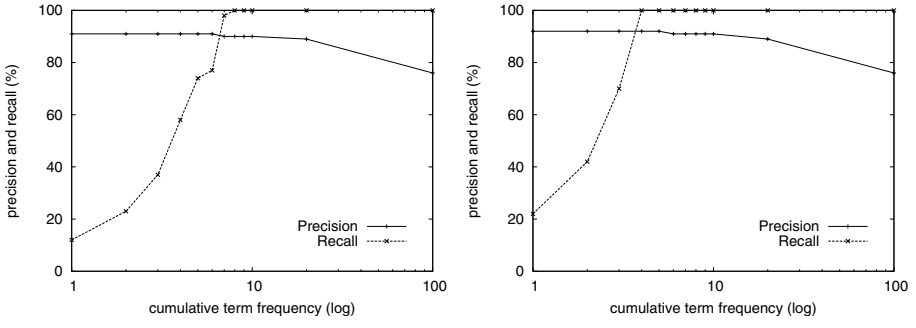
Algorithm 2. $LevelwiseBreadthFirst(G, t, l, state)$

Input: $G_{f,k}$ a filtered co-occurrence graph, a tag $t \in V$, a threshold level for shortest path l, the current *state*
Output: a set of nodes.
if $state < l$ **then**
  $Temp_s = Temp_s \cup \{Tp(s, r)$ as set$\}$. {get the top-r neighbors nodes of s at distance 1.}
  for all $t_1 \in V$ in $Temp_s$ **do**
    $LevelwiseBreadthFirst(G, t_1, l, state + 1)$
  end for
end if



(a) (b)

Fig. 1. (a) Upper and lower bound of the average tag frequency by varying top-r most important neighbors, (b) misspelled tags (%) by varying cumulative edge frequency



(a) (b)

Fig. 2. Precision and recall (%) by varying the tag frequency (log). (a) $r = 10$, $l = 1$, $\delta = 1$, $k > 0$, (b) $r = 10$, $l = 2$, $\delta = 1$, $k > 0$.

weights increases. This means that by removing edges with an associated weight less than k, the percentage of misspelled tags (not isolated nodes) in the whole set decreases.

Figure 2(a) shows values for precision and recall by varying the threshold on tag frequency. Such a threshold works as a filter on low frequency tags. This evaluation uses only the first level of neighbors of tags ($l = 1$). Precision is high (up to 90%), and recall improves significantly (up to 100%) putting a threshold on the tag frequency to values close to 10. On the other hand, by fixing such threshold to values greater than 20, our method starts losing precision. Figure 2(b) shows values for precision and recall by varying the threshold on tag frequency when the evaluation uses two levels of neighbors of a tag ($l = 2$). Precision is still high (up to 90%), while recall improves significantly (up to 100%) by putting a threshold on tag frequency to values close to 4.

4 Conclusions

We presented a tag spelling correction method exploiting a graph model representing co-occurrences between tags. Tags from YouTube's resources are collected and represented on a graph. Such a co-occurrence graph is then used in combination with an edit distance and term frequency to obtain a list of right candidates for a given possibly misspelled term. Experiments show that this collaborative spell checker yields a precision up to 93%, with a recall of 100% (in many cases). We plan to extend this work by considering not only co-occurrence of tags within the same objects, but also to consider the effect of neighborhoods at various distance levels.

References

1. Silvestri, F.: Mining query logs: Turning search usage data into knowledge. Found. Trends Inf. Retr. 4, 1–174 (2010)
2. Surowiecki, J.: The Wisdom of Crowds. Anchor (2005)
3. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. 24, 377–439 (1992)
4. Cucerzan, S., Brill, E.: Spelling correction as an iterative process that exploits the collective knowledge of web users. In: Proc. EMNLP (2004)
5. Whitelaw, C., Hutchinson, B., Chung, G.Y., Ellis, G.: Using the web for language independent spellchecking and autocorrection. In: Proc. EMNLP 2009. ACL (2009)
6. Schaback, J.: Multi-level feature extraction for spelling correction (2007)
7. Merhav, Y., Frieder, O.: On multiword entity ranking in peer-to-peer search. In: Proc. SIGIR 2008. ACM, New York (2008)
8. Ahmad, F., Kondrak, G.: Learning a spelling error model from search query logs. In: Proc. HLT 2005. ACL (2005)
9. Freund, J.: Mathematical Statistics. Prentice-Hall, Englewood Cliffs (1962)