Use Cases for NLP and IR in Tiki

Table of contents

What's this about?

This page describes different Use Cases for Natural Language Processing (NLP) and Information Retrieval (IR) functionalities in Tiki.

A lot of the ideas on this page relate to the more general concept of Content-Aware, Intelligent Tiki. By this, we mean a Tiki that is somehow "aware" of what it is about, can help its users is an "intelligent", human-like way.

Most of these were generated at TikiFest2013-Gatineau-NLP and TikiFest2013-Montreal-NLP.

2013-06-03 Presentation about Natural Language Processing (NLP) in Tiki

in French, with Alain Désilets, Matthieu Hermet and Nicolas Rosenfeld (filmed by Marc Laporte)

Use Cases

We came up with a great number of Use Cases, which fell into 3 clusters below.

Eliminate Duplicates

Large Tiki sites like *.tiki.org, or large corporate wikis, often contain pages that are close duplicates. Often a person creates a page or uploads a document, without realising that someone already has created such a page or document.

These kinds of duplicates can be eliminated either at the moment they are created (ideally), or after the fact.

Use Case: "The page you just created may be a duplicate of these"

When a user saves a new page for the first couple of times, the system looks for the pages that are most similar to it. If some of them are too close for comfort, it signals that to the user so he can merge his new page with an existing one if needed.

Use Case: "The document you uploaded may be a duplicate of these"

Very similar to the previous use case, except that it applies to uploaded documents.

Duplicate elimination may be even more useful for Uploads, because Uploaded files are more "opaque" and there are more chances that you will create a duplicate. By "opaque", I mean that it's harder to search through the content of uploads, or view summaries of them. So it's more likely that you will not realize that the content you are about to upload is already there.

Use Case: "Show me the most similar pages that match query X"

This is for eliminating duplicates **after the fact**. Say, you want to check for duplicates in the documentation about blogs. You go to doc.tiki.org, and ask to see the most similar pairs of pages that contain the word "blog".

You can then inspect the pairs one by one, starting from the top, to see if any of them might need to be merged.

Find similar or related to a particular interest

Given that I am interested in certain things, I want the system to show me other similar things that might be of interest to me.

Use Case: "See also"

When reading a page|article|tracker|upload you get a module that says "See also". It points to a short list of the pages|articles|trackers|uploads most similar to the one you are looking at.

Use Case: "You might be interested in the following job postings|resume"

Marc mentioned a customer that allows people to post resumes or jobs. When someone enters his resume, it would be nice if they saw a list of job postings that most closely match their resume.

Conversely, when an employer posts a job, it would be nice if he saw a list of the resumes that most closely match the job.

Use Case: "You might want to meet the following people"

When a new user fills his/her user profile, it would be nice if he/she saw a list of the users with the most similar profiles.

Use Case: "People who bought this product also bought these"

When using Tiki for e-commerce, it would be nice if we could do this.

Use Case: "The product you just purchased is very similar to the following"

Very similar to the above, except that the similarity is based on the actual product description, as opposed to the social use of the products (i.e. who purchased what).

Use Case: "People who visited this page also visit those"

Very similar to the "See also" use case, except that here, the similarity is based on Social Use of the pages (who looks at what) rather than the content of the page.

Use Case: "The following people seem to visit|like the same kinds of pages as you"

Similar to the "You might want to mee the followig people", except that affinity is not based on the content of the user profiles, but rather on their usage of the site (what pages they visit).

Use Case: "The following groups|pages might want need to meet|reference each other"

In large organizations, it's not uncommon for two groups to be working on similar things without knowing about each other.

It might be nice if you could identify groups of people or pages which are very similar, but are not "aware" of each other. In the case of pages, awareness means referencing each other, whereas for groups of people, it might mean to be part of a common category or users group.

Use Case: "You might want to supplement your keywords with those"

Whenever you enter keywords to describe a thing in tiki, the system could suggest additional keywords that seem similar or related to them.

Use Case: "New content of interest to you has been added"

The user identifies several topics of interest. Each topic is described by a series of pages, blog posts, etc... that are on that topic.

Then, once a day, the system finds all the pages that are similar to the pages in a given topic. If there is a similar page that was not in the list produced the day before, it notifies the use that there is a new page of interest for him on the site.

Use Case: "The following Tiki sites near you might be of interest to yours"

This is in the context of TikiConnect. Basically, when you create a new Tiki site, you can opt into TikiConnect. This means that your site will be sending some information to the Tiki "Mother" site, and the mother site will be able to find connections between your site and other similar sites.

Use Case:"Show me some pending bugs that I might be able to fix"

When a user goes to the dev.tiki.org with the intent of helping to fix some bugs, instead of manually looking for bugs that seem to be down his alley, he can click on a button and the system automatically suggests bugs that seem to fit his skills.

The way to decide whether a bug fits the user's skills, could be based on:

- Bugs he has fixed in the past
  - Bugs whose description is similar to the description of bugs he fixed in the past, are flagged as being a good fit.
- A description of his Tiki Skills
  - User expresses those skills in his profile (could be freeform text or a more constrained tracker). Bugs whose description is "similar" to that skills profile are tagged as being a good fit.

Use Case:"Which items should I read in my news/RSS feed?"

Say you are using Tiki to allow a team of people to collaborate on a technology watch. They use Tiki to syphon pages from given RSS feeds, and then they pick pages, read them, annotate them, forward them to each other, etc...

In a context like this, it would be nice if Tiki was able to do a first sifting of the vast amount of data that comes from the Feed, and only show you a short list of 20 items that most closely resemble the items that you have found to be interesting in the past.

Also, when you are doing technology watch, there are usually sub-topics that you are interested in. It would be nice if Tiki could automatically put items of interest in the proper "bin".

Organizing content

Often, you want to organize content by assigning it categories, or by splitting it into mutually exclusive clusters.

Use Case: "Split conference participants into tables of 10"

Say you are using a Tiki to organize a conference. Each participant filled a user profile. You want to set the seating arrangement at lunch, so that people with similar interests will be at the same table. A sheet would be printed out at each table, with list of participants and their keywords. Thus, the conversations will be very interesting. "I see Bob is interested in X and Y. So am I! Which one of you is Bob?".

The event lasts 3 days so the system would sit you with new people every day.

Use Case: "Split work to do on this wiki among 15 volunteers"

Say you have 15 people who volunteer to cleanup doc.tiki.org.

Would be nice to split the whole site into 15 cluster of approximately the same size, with each cluster corresponding to the interests of a particular volunteer.

Use Case: "Where should this go?"

You create a new page and you want to know which category to put it in. When you save it, the system automatically suggests the most likely categories.

Gisting/Summarising content

Often you may want to get a feel for what's contained in a one or more documents. Use cases in this section deal with this kind of situation

With IR methods, you can typically get 2 kinds of automatic summaries:

- Key phrases
  - These short (1-3 words typically) expressions that appear "surprisingly often" in the documents.
- Key sentences/paragraphs
  - These are sentences and paragraphs that contain a "surprisingly large number" of keyphrases (as defined above).

Use Case: "What are the most frequent questions people ask here?"

Useful for starting a FAQ list.

You point the system to the root of your site, or to a forum, a category, etc..., and it automatically extracts a list of questions people have asked frequently.

You can then use this to start a FAQ list.

Use Case: "What NEW questions are people asking?"

Useful for augmenting an existing FAQ list.

You:

- point the system to a source of documents (ex: a forum, a category)
- point the system to a source of FAQ (ex: a page, a category)

The system automatically extracts questions that are contained in the documents source, which don't seem to be related to the questions in the FAQ.

Use Case: "What do people talk about on this site?"

When users come to the site, they see a list of keyphrases on the right.

This is very similar to the module that displays the most common freetags, except that you don't have to rely on users to create tags. The tags are created automatically by the system.

Use Case: "What does this category talk about?"

When you click on a category, you can see a a "summary" of everything in that category, in the form of a list of automatically extracted keyphrases.

Use Case: "What do people say about X on this site"

When user does a search for X, he gets a list of hits. But at the top of the hits page, he also sees a list of keyphrases extracted automatically from the content of all the hits.

There is also a link "Summarize hits". When the user clicks on it, he gets a list of key sentences and/or paragraphs extracted from the content of all the hits.

Use Case: "What major themes were discussed in my interviews"

Say you are a researcher like Regis, or a marketing researcher who use a Tiki site to transcribe interviews or focus groups.

Once you have transcriptions, you want to get a feel for what the major themes were that people talked about in those interview/focus groups.

Use Case: "What are people posting on my site during my event"

Say you are running a 2-day conference, and you want to have some breakout sessions on the second day. You want the topics for the breakouts to emerge naturally out of what people post on the site.

You have a wiki page for each talk and people can post comments during the talk, or before or after.

Now you want to have a short list of the 5 broad topics that people have talked most about in the comments.

This can be either in the form of:

- The 5 automatically extracted keyhprases from the comments
- The 5 automatically generated clusters of comments

A variant of this idea is to provide a tiki "channel" through which people can tweet about your event, but going through Tiki, so that Tiki knows what people are tweeting about. Or, have a way to siphon from Twitter, all the posts with a particular hashtag.

Feeling the "mood"

Sometimes you are not so much interested in what people are saying, but rather, the mood they are in when they are saying it.

There are several algorithms out there that can automatically evaluate the mood expressed in a particular text. For example, the degree to which the text expresses positive or negative sentiment.

Below are several Use Case that show how this could be used in Tiki.

Use Case: "Are people in my community happy or unhappy right now?"

You click on a button, and the system displays a timeline that shows the evolution of the mood on your community in say, 1 week increments. The mood of each period is caculated based on the content of contributions for that period (either new content, or modifications to existing content).

You can ask the system if there has been a significant shift in mood (either positive or negative) in the last N weeks.

Use Case: "Which of my customers are most unhappy?"

Say company X uses a Tiki site to interact with their customers.

You want to know who, among the customers who are posting the site, is most unhappy, so you can maybe do something to make them happy.

Use Case: "Which of my products are customers most happy/unhappy about?"

Say company X uses a Tiki site to interact with their customers about a series of products. The site is organized in such a way that content that relate to each product resides in a different category.

You click on a button, and the system ranks each of the products in terms of how positive/negative customers feel about it (as expressed by the content they have contributed to the site).

Use Case: "Show me the most positive/negative things people have to say about X"

Say company A uses a Tiki site to interact with their customers about a series of products. The product manager for product X wants to see the most negative/positive thing people customers have gto say about his product.

He starts a query for X, and asks to see the most positive/negative pages/paragraphs/sentences that mention X.

- Sorts them in order of positive/negative feeling

Use Case: "Who are the positive/negative forces in my community?"

You are moderating a large community of users. You want to know who are the trolls on your site, i.e. people who only say negative things, or who piss people off and cause reactions that have a lot negative feel to them. Once you have identified them, you may register to be notified of everything they post, so that you can intervene and moderate conversations when they start to fester.

Conversely, you want to know who are the positive forces in the community. Those are people you want to stroke and encourage to stick around.

Use Case: "Tell me when a conversation needs moderation"

As a moderator for a large community, you may be too busy to read each and every post and thread. So it might be nice to be notified automatically when a particular conversation seems to be turning into a flame war.

Infrastructure needed

What would we need to support the above use cases?

More like this (mlt) functionality

Most of the use cases assume that, given a particular "thing" in wiki (a page, a tracker, a user profile, etc...), we are able to find the most similar things in the site.

So we need a class that can do this. It might be nice to have it be a plugin so it could be embedded in wiki pages.

The similarity metric used by this mlt plugin may be based on the actual content of the thing, or based on how people in the community "use" that thing (i.e. visit, modify, purchase, like, etc...).

It would be nice however if this kind of social data could be codified as a kind of "pseudo-content" field associated with the "thing". That way, doing mlt based on social use would just involve telling the mlt plugin to use a particular field of the thing to compute similarity, and it wouldn't have to know that this field is not actual content, but rather meta data about social use of the thing.

Is seems that Elasticsearch already has a mlt functionality. We should take advantage of that.

As of 2013-06-06, Mathieu Hermet is looking into this.

More like THESE functionality

Some of the Use Cases involve classifying or clustering **groups of documents**. In those cases, we don't just want to find individual things that are similar to a given individual thing. Rather, we may want to find things like:

- groups of things similar to a given individual thing
- individual things similar to a given group of things
- groups of things similar to a given group of things

As far we know, Elasticsearch does not have such a "More like THESE" functionality, but it should be possible to build on top of the "More like this" functionality to create it.

For example, you might be able to create pseudo-documents for each group and index them with Elasticsearch. The content of the pseudo-document would be the concatenation of the content of each of its members.

Note: This may be inefficient for large groups. For example, if you have a group of 1000 documents, then adding a single document would mean:

- Retrieve the content of those 1000 documents
- Concatenate them with the new document to be added
- Reindex this pseudo-document

But as a first pass, it should do the trick. For larger groups, we may have to plug into Elasticsearch at a lower level, to be able to update a group's word frequency model incrementally.

Automatic clustering algorithms

Some of the use cases assume that you can split an otherwise unorganized bag of things, into clusters.

I don't think Elasticsearch has that. However, it may be that something like Maui can do it, and plug on top of Elasticsearch.

Semantic augmentation of content

The Elasticsearch mlt functionality may not work that well with short texts (ex: user profiles), because the chances of having overlapping terms between such texts is lower.

But it could be that we can augment the terms present in such short texts, with terms that are somewhat "implied". This can be done with an Explicit Semantic Analysis framework (which is also included in Maui apparently).

Available NLP and IR toolkits

NLP and IR kits

- Rubix ML has been added to Tiki23
- Elasticsearch (Java)
- Maui (Java): Automatic keyphrase extraction
- GATE (Java)
  - JAPE: Some kind of pattern matching language for GATE
  - ANNIE: Not sure what this is, but it runs on top of GATE...
- Apache UIMA
  - http://uima.apache.org/
  - Description from site: Our goal is to support a thriving community of users and developers of UIMA frameworks, tools, and annotators, facilitating the analysis of unstructured content such as text, audio and video.
- MALT (Java)

- MELT (Python)
- OpenNLP (Java)
- NLPTools (php):
  - http://php-nlp-tools.com/
  - Claims to support Spam detection, Sentiment detection, Part of Speech Tagging, Classification and Clustering, Similarity
  - Seems much less mature than the Java frameworks.
- UClassify (cloud-based)
  - http://www.uclassify.com/
- Apache Mahout
  - Our core algorithms for clustering, classfication and batch based collaborative filtering are implemented on top of Apache Hadoop using the map/reduce paradigm.

Usage patterns kits and algorithms

Many of the scenarios on this page are not so much about NLP nor IR. They are more about analyzing usage patterns like: visits to pages, recommendations, etc...

Below is a list of toolkits and algorithms for doing that sort of thing:

- Slope 1
  - http://en.wikipedia.org/wiki/Slope_One
  - A really simple (almost trivial to implement) algorithm for Social Filtering.

What to implement first

We should start by going for Use Cases that

- Have a high potential value
- Yet, are implementable with barebone Elasticsearch mlt functionality

Examples include:

- Functionality for identifying possible duplicates
- "See also" Use Case, at least the one that is based on the actual content of the page, not on the social use of it.

Pictures

Brainstorming on params to apply to each use case

- http://www.elasticsearch.org/guide/reference/query-dsl/mlt-query/
- Did you mean?
- http://www.elasticsearch.org/blog/you-complete-me/