Planning a Reverse Thematic Dictionary

Tara Edwards
United States of America

Abstract

In this paper I propose a plan on how to write an application that generates proper names from digital corpora. The application uses constructed rules and keywords lists describing Julian months. The corpus is stored on the web and in electronic text, with physical books used as proofreading references. The constructed naming convention keeps name themes by these rules: to have as many names I output as possible, to be drawn from specific languages, and to rank output to display. The application is an electronic dictionary with a relational database acting as a map. The database would map one keyword to a set of co-occurring terms. Terms used would be words classified by WordNet as physical or abstract entities. Each of the terms would map to a list of names, filtered by gender. Output would be ranked by collocation distance from keyword appearing in shared text spaces. The first method is searching for a set of keywords in documents. Searching the corpora for entries, the reader would expect collocations with the chosen keyword with additional sets of keywords. To test this idea, I filtered common English words and sets of keywords from corpora containing keywords. The output generated new possible terms to be mapped to names.

* * *

Inspiration

The application is conceived as part of the planning for a franchise that includes a serial fiction story, a roleplaying game, and an audio drama. I am planning the application and the franchise. I knew before starting that I ran the risk of not finishing both projects. Either project would require time, funding, and other people. At least, I can document planning the application.

I am grateful to ICOS and editor Ruth Feiertag for encouraging me to document the project. I hope this report interests other writers, parents, onomasticians, and computational linguists. I hope they put it to good use. Maybe someone will finish what I have started.

The serial would range from the 16th century to 21st century, and would take place in many countries and use a variety of languages. The family at the center of the stories runs an inner cult influenced by cross-cultural beliefs about the supernatural. During this time spread, the story's family develop a complex family naming convention. Unknown to outsiders, these names describe rank and duties in their family cult. In the story, the main characters give offspring names related to calendar months. The meanings of the names are not just months. The meanings are also related to birthstones, holidays and other monthly events, the origin of month names, and translations in other languages. Thus, the abstract entity of time is connected to physical entities, to put it in WordNet categories. These names will be used in the stories, and stored in the application database to be chosen by their meanings.

Names and Their Environment. Proceedings of the 25th International Congress of Onomastic Sciences, Glasgow, 25-29 August 2014. Vol. 4. *Theory and Methodology*. *Socio-onomastics*. Carole Hough and Daria Izdebska (eds)
First published 2016 by University of Glasgow

Our program will give me and other people interested in these stories the resources to locate or create names that conform to their characters' complex naming rituals. Also, the names can be expanded by searching for names with those meanings.

The origins of the names can be from any language. The application, however, places bounds on selecting names. One constraint the application imposes is that certain language origins are given priority over all other languages. These languages are: Arabic, Aramaic, Cornish, French, German, Greek, Hebrew, Irish, Hindi, Italian, Latin, Manx, Portuguese, Sanskrit, Scottish, Spanish, and Welsh. If needed, the application can expand to Danish, Russian, Turkish, Chinese, Japanese, Armenian, Dutch, and Maori. Definitions of names are in English.

At least one list of month names can be found on the web, from 20000-names.com. Each month has at least one name. However, the list of 12 to 24 names is not enough for an old family.

Writers give names based on a theme. They do it to create a sense of unity and cohesion within their work ('Theme Naming', TV Tropes). The writer determines how many and how varied the character names will be.

A writer has many books and websites to help with naming characters. A writer can choose from forward and reverse dictionaries of first names. The writer can generate random names or search for names by keyword. However, no tool maintains consistency in character name themes. No tool has the cohesion of a name list and depth of a first name reverse dictionary. Also, no tool stores multiple results in one database, and analyzes them for consistency with the time and place of setting. Not every writer is concerned with that. Writers who are concerned wish to spend more time writing than worrying.

A computer application seems to be an affordable way to find and use names for this fictional family. An application would store names collected by multiple sources. The effective application will not be a list of names, but a dictionary through which one would be able to find and display names based on user input. Since no writer sticks to one theme, the application will eventually rely on user input. Since themes are more than one keyword, the search function needs to use more than keyword search.

To build this application, anyone interested can use a reverse name dictionary, a program for a reverse dictionary, and an open source programming tool kit with support for corpora and natural language processing algorithms. Since a reverse dictionary requires a forward dictionary, my research includes instructions for dictionary planning.

Shaw Approach

Ryan Shaw and other researchers at Google designed a reverse dictionary as a relational database acting as a map. The database would map one keyword to a set of co-occurring terms. If results were below the threshold, the application would expand the user inquiry based on semantic similarity of the search terms. The application would search through a complex network of lexical relations. It would find other synonym sets linked to the keyword (Bird and Klein 2009: 71).

Shaw et al. (2013) worked on a project to solve what is called the concept similarity problem. If words were put in a hierarchy, the words on the top would describe the most

abstract concepts. From there, words are connected downward to the most specific terms. The best known example of such a hierarchy is WordNet. The challenge would be to find the shortest path between words. Shaw used WordNet, and so will I. Designers often model these concepts as single words. However, concept phrases can have similar paths, but have no exact matching words.

The next obstacle is that building a vector space, with its collection of elements and calculations, requires large storage space, and a lot of processing time. However, the application needs to calculate the combined probability of individual words in user input while the application runs. The application then needs to compare that output to stored probabilities of words found in dictionary definitions.

One concern not shared with Shaw is the size of text collection, or corpora. The ability to identify meanings of words in context in a computational manner (Yarowsky), or word sense disambiguation, depends on corpora. Shaw *et al*. (2013) limit corpora to dictionary definitions of 50-100 words. I have collected information from Wikipedia and other websites, and stored them in spreadsheets and text documents. Natural Language Toolkit (NLTK) has plain text corpora and access to WordNet. I will use these available resources.

Book Approach

Two reverse dictionaries are in print. *First Name Reverse Dictionary* by Yvonne Navarro (2007) is organized first by gender, then by headword in alphabetical order, then by phrases related to headword, then by names and their variations. In the index of names, a name can match with more than one headword. *Baby Names Made Easy* by Amanda Barden (2009) is organized by theme, then gender, then names in alphabetical order.

I chose another book, a reverse dictionary of symbols listed by subject. Daniel Olderr (1992) ordered subjects alphabetically, then by method (like heraldry), then by symbol in alphabetical order. Appendix 1 contains the entries for English language months. Entries from Olderr (2009) match 20,000 names: months can be related to gemstones, animals, plants, activities, seasons, and monthly events.

Dictionary Planning Steps

Before I will write the application that generates proper names from digital corpora, I will plan how to write that application. The process would begin with the collection of material, then move to filling up the entries. Next would come the compilation of a word list, and finally deciding on the structure of the dictionary entries. The entry will consist of description and definition of meaning, script, and pronunciations. Since I had gathered material beforehand, planning felt like going backwards. I had to think of how potential users would ask for names, as well as how to organize the names.

The project’s scope size is unknown. If potential users or sponsors are more interested in the application than in the stories, the scope may be limited to how it works rather than what material to use. Its purpose is to get most likely names for the exact phrase or words

associated with it. We expect its primary users to consist of parents and fiction writers looking for appellations to bestow − or inflict − upon their progeny and creations. The range of coverage will be determined by name origins, information the users are seeking, and how much computer space I have. Since NLTK makes source code freely available and allows modifications and redistribution, I would share rights to the code with other users.

The method of collection is a two-step process. I will code text extraction programs with NLTK in Python. Once the programs gathered enough data, I will verify the data with print dictionaries on my bookshelves. This is to compensate for possible inaccuracies in web sources.

The type of dictionary is an electronic application with a database. I will use NLTK to create the word list, by sorting and filtering for names and unusual words.

Website Approach

I found two examples of baby-name site maps. They will be compared to other baby name websites. 20000-names.com has lists by origin and lists by theme. The *Behind the Name* website, like most sites of this kind, has lists arranged by themes, lists organized by origin, and lists determined by alphabetical order. Many other baby websites have search engines and filters. However, no website has searching abilities beyond keyword searches in meanings, or beyond specifying gender, origin, or alphabetical letter. No website has the format of theme represented by a headword, phrases with the headword, and then names.

Other websites provide potential names and associations. The *Gazetteer of Planetary Nomenclature* has names of the moons and geological features of the planets. Since months can be associated with planets, this is a rich source of specific information.

Another online resource is the HTML version of *Star Names and Their Lore*. This includes corrections to Allen and Allen (1963) as needed. In the original and HTML version, bolded or colored words are proper names, including translations of constellation names, stars within the constellation, and asterisms. The text has months when constellations can be seen, which matters when generating more names for months.

If needed, other websites have exact seasonal and holiday associations of the months. This information would expand on how family members are ranked, and roles they play.

NLTK Approach

The first function I created searches for a set of keywords in documents. Searching the corpora for entries, I expected the chosen keyword would be found with additional sets of keywords. For the results to be useful, there had to be more than one instance of a non-common English word.

I included these considerations to keep the function on track. Words and punctuation are separated and tokenized by an NLTK function. The function filters by converting results to lower case alphabetical characters. It then divides names corpora available from NLTK into male and female names. The function then finds differences between the corpora and a list of English words. Words found in the former, but not the latter make up unusual words

list. After that, the function calculates frequency distribution of the set of words. Appendix 2 has the results sorted alphabetically.

Results of the Program

To test this idea, I filtered common English words and limited sets of keywords from corpora containing those sets. Along with the English words list, I added an English language stopwords list from NLTK. One test was on the text documents. The other test was on text and comma separated values (CSV) documents. Probability from only text and from text and CSV were added together. The results produced words with small probabilities, as I intended. As the project progresses, the output generated may be used to introduce new possible terms. The found terms may function as an temporary index to mapped names.

In writing the complete application, I (or another researcher) would keep in mind what the output should look like. First, the month would be printed, then a list of names. These names would be based on non-stop-words in meaning found close by the chosen keyword (month or otherwise). The names are then listed by origin, gender, and definition. These conditions would prevent empty lists. If the potential list is too small, the application would find names with words collocated with the keyword. The application would continue until the list has at least five names. Name variations may be listed separately or in a field connected to the name.

If need be, the application will use data already in documents as training data. Training data can also come from entries from Olderr, Barden, and Navarro. The application will then use data extracted from websites as testing data. Testing data might also come from Wikipedia data dumps, other corpora, search results, and full websites. I or some other researcher may use one of the listed methods or a combination of them.

While anyone adding to the database can use contemporary text, they may also use older texts to find other keywords and verify definitions. The application will use most common spellings of a name, its variations, and diminutives.

Further Work

Next steps are: reconciling different ideas on algorithms; deciding on relaxing word association restrictions; deciding on sorting functions, be it by name, by keyword, by gender, or by association.

The electronic dictionary was to use constructed rules and keywords lists describing Julian months. I had several spreadsheets detailing objects and concepts associated with the months. However, given available corpora, it may be possible to fill keywords without a separate database. I would need other mechanisms to ensure precision after I took care of recall. Anyone developing the application will use this presentation as a resource for decisions he or she will make. If others are interested, I can share earlier documents on Google Drive.

I hope if the application is finished, anyone will understand the motivations and actions of the characters, whether they are reading, listening to, or playing out the stories.

I also hope the application will help other writers by making the character naming process seamless enough so the reader can enjoy their characters' stories.

Tara Edwards
Independent scholar
United States of America
taedwards@gmail.com

References

Allen, R. and Allen, R. (1963) *Star Names: Their Lore and Meaning*. Rev. edn. New York: Dover Publications.

Barden, A. (2009) *Baby Names Made Easy: The Complete Reverse Dictionary of Baby Names*. New York: Simon & Schuster.

Bird, S. and Klein, E. (2009) *Natural Language Processing with Python*. Beijing: O'Reilly.

Planetary Names: Planet and Satellite Names and Discoverers (2005) Date of access: 10.08.2014. Available online at: http://planetarynames.wr.usgs.gov/Page/Planets

Navarro, Y. (2007) *Complete Reverse Dictionary of Names*. 2nd edn. Jefferson, N.C.: McFarland.

Navigli, R. (2009) 'Word Sense Disambiguation: A Survey'. ACM Computing Surveys 41.2. 1-69.

Olderr, S. (1992) *Reverse Symbolism Dictionary: Symbols Listed by Subject*. Jefferson, N.C.: McFarland.

Pattanayak, D. (n.d.). 'Dictionary Making Phase: Preparation.' Central Institute of Indian Languages. Date of access: 10.08.2014. Available online at: http://www.ciil-ebooks.net/html/lexico/link7.htm

Shaw, R., Datta, A., Vandermeer, D. and Dutta, K. (2013) 'Building a Scalable Database-Driven Reverse Dictionary.' *IEEE Transactions on Knowledge and Data Engineering* 25.3. 528-540.

TV Tropes (2011) 'Theme Naming'. Date of access: 9 August 2014. Available online at: http://tvtropes.org/pmwiki/pmwiki.php/Main/ThemeNaming

Appendix 1

Olderr (1992) has entries for the months.

April	cherry blossom	June	alexandrite
April	cuckoo	June	chalcedony
April	hyacinth	June	Juno
April	rook	June	crab
April	diamond	June	harvest
August	harvest	June	harvesting
August	harvesting	June	agate
August	august, majestic	June	alexandrite
August	Augustus	June	pearl
August	peridot	June	pomegranate
August	sardonyx		blossom
August	pear blossom	June	white flower
December	ten	March	Mars
December	goat with spiral tail	March	heliotrope
December	turquoise	March	ram
December	zircon	March	sheep
December	poppy	March	aquamarine
February	Februus	March	bloodstone
February	amethyst	March	tree peony
February	peach blossom	May	robin
February	heron	May	Maia
January	Janus	May	emerald
January	garnet	May	magnolia blossom
January	plum tree	May	hawthorn
July	harvest	November	nine
July	harvesting	November	topaz
July	lion	November	gardenia
July	ruby	November	sower
July	star ruby	October	eight
July	lotus	October	aquamarine
July	Julius	October	opal
July	young, downy	October	tourmaline

October	scorpion
October	chrysanthemum
September	balances
September	sapphire
September	turquoise
September	chrysanthemum
September	mallow blossom
March	Pisces
March	Aries
April	Aries
April	Taurus
May	Taurus
May	Gemini
June	Gemini
June	Cancer
July	Cancer
July	Leo
August	Leo
August	Virgo
September	Virgo
September	Libra
October	Libra
October	Scorpio
November	Scorpio
November	Sagittarius
December	Sagittarius
December	Capricorn
January	Capricorn
January	Aquarius
February	Aquarius
February	Pisces

Appendix 2

Results of NLTK experiment

agate	0.000701754385965
air	0.00210526315789
alexandrite	0.000701754385965
amethyst	0.00140350877193
aquamarine	0.00140350877193
archer	0.000701754385965
army	0.000701754385965
aster	0.00140350877193
autumn	0.00210526315789
balance	0.000701754385965
bearer	0.000701754385965
best	0.000701754385965
black	0.00140350877193
blaze	0.000701754385965
bloodstone	0.00140350877193
body	0.000701754385965
born	0.000701754385965
breeze	0.000701754385965
brook	0.000701754385965
bull	0.000701754385965
burn	0.000701754385965
calendula	0.000701754385965
calf	0.000701754385965
carnation	0.00140350877193
carnelian	0.000701754385965
carrier	0.00140350877193
cat	0.000701754385965
cattle	0.00140350877193
centaur	0.000701754385965
center	0.000701754385965
chariot	0.000701754385965
chrysanthemum	0.000701754385965
chrysolite	0.000701754385965
cliff	0.000701754385965
constellation	0.000701754385965
cosmos	0.00140350877193
cow	0.00140350877193
crab	0.000701754385965
daffodil	0.000701754385965
daisy	0.00140350877193
delta	0.000701754385965
diamond	0.00140350877193
eagle	0.000701754385965
earth	0.00280701754386
emerald	0.00280701754386
ewe	0.000701754385965
fire	0.00210526315789
fish	0.000701754385965
flame	0.000701754385965
form	0.000701754385965
fowl	0.000701754385965
garnet	0.00140350877193
girl	0.000701754385965
gladiolus	0.00140350877193
glory	0.000701754385965
goat	0.00140350877193
gracious	0.00701754385965
green	0.000701754385965
guessing	0.000701754385965
guru	0.000701754385965
hawthorn	0.000701754385965
hill	0.000701754385965
holly	0.000701754385965
honeysuckle	0.000701754385965
horse	0.00140350877193
hunter	0.00140350877193
ice	0.000701754385965
jasper	0.000701754385965
jonquil	0.000701754385965
killer	0.000701754385965
kin	0.00280701754386
known	0.00210526315789
l	0.00140350877193
lake	0.000701754385965
lamb	0.00140350877193
larkspur	0.000701754385965
life	0.00210526315789
lily	0.00280701754386

lineage	0.000701754385965
lion	0.000701754385965
lotus	0.00140350877193
maiden	0.00210526315789
mare	0.000701754385965
marsh	0.000701754385965
moonstone	0.000701754385965
morning	0.000701754385965
mother	0.000701754385965
mou	0.00631578947368
mountain	0.000701754385965
mum	0.000701754385965
narcissus	0.00140350877193
night	0.000701754385965
notorious	0.000701754385965
ocean	0.00210526315789
onyx	0.000701754385965
opal	0.00140350877193
oread	0.000701754385965
patron	0.000701754385965
pea	0.000701754385965
pearl	0.00140350877193
peridot	0.000701754385965
phoenix	0.000701754385965
pony	0.000701754385965
poppy	0.000701754385965
preceptor	0.000701754385965
primrose	0.00140350877193
rainbow	0.00140350877193
ram	0.000701754385965
redeemer	0.000701754385965
river	0.000701754385965
rock	0.000701754385965
rose	0.00140350877193
ruby	0.00210526315789
sapphire	0.00210526315789
sardonyx	0.00140350877193
savannah	0.000701754385965
scales	0.000701754385965
scorpion	0.000701754385965
sea	0.00350877192982
seller	0.000701754385965
sheep	0.00140350877193
snake	0.000701754385965
snow	0.00210526315789
snowdrop	0.000701754385965
soil	0.000701754385965
son	0.000701754385965
spouse	0.000701754385965
spring	0.00210526315789
stallion	0.000701754385965
star	0.000701754385965
steppe	0.000701754385965
storm	0.000701754385965
stream	0.000701754385965
summer	0.00280701754386
summit	0.000701754385965
sunshine	0.000701754385965
supine	0.000701754385965
sweet	0.000701754385965
tempest	0.000701754385965
topaz	0.00140350877193
torch	0.000701754385965
tourmaline	0.000701754385965
turquoise	0.00140350877193
twin	0.000701754385965
universe	0.000701754385965
upon	0.000701754385965
valley	0.00210526315789
violet	0.00210526315789
virgin	0.000701754385965
water	0.00631578947368
waterer	0.000701754385965
well	0.000701754385965
white	0.000701754385965
wind	0.000701754385965
wine	0.00631578947368
winter	0.00210526315789
zeus	0.00210526315789
zircon	0.000701754385965