School of GeoSciences

Dissertation

For the degree of

MSc in Geographical Information Science

Student Name: Ciaran Hoy

Date: August 2016
Copyright statement

"Copyright of this dissertation is retained by the author and The University of Edinburgh. Ideas contained in this dissertation remain the intellectual property of the author and their supervisors, except where explicitly otherwise referenced.

All rights reserved. The use of any part of this dissertation reproduced, transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a retrieval system without the prior written consent of the author and The University of Edinburgh (Institute of Geography) is not permitted."

Statement of originality

"I declare that this dissertation represents my own work, and that where the work of others has been used it has been duly accredited. I further declare that the length of the components of this dissertation is 5087 words for the Research Paper and 1517 words for the Technical Report."

Signed ___________________________ Date ___________________________
"Ecoparsing case study: modified geoparsing to extract tiger sightings from 19th century travel diaries."

Abstract

This paper will explore the use of modified geo-parsing techniques to extract data for spatiotemporal incidences of ecological significance. For simplicity, this process will be referred to as ‘eco-parsing’. To illustrate this concept, tiger sightings will be extracted from a corpus of 19th century English language travel diaries for Asia. A prototype eco-parsing tool was developed with a modifiable automated classification systems to promote more probable and relevant sightings to the user for manual resolution. Eco-parsing is method to rediscover underutilised information and bring it to modern use by applying appropriate data standards. The semi-automatic identified spatiotemporal incidences can be mapped and the data extracted for further analysis. The technique is shown to be successful, with some limitations, and the need for further development is demonstrated.
# Table of Contents

- **Introduction** .......................................................................................................................... 5
- **Research questions** .................................................................................................................. 5
- **Applications of text parsing** ..................................................................................................... 6
- **The Semantic Web** .................................................................................................................... 7
- **The potential the ecological value of historical texts** ................................................................. 7
- **Eco-parser tool functionality** .................................................................................................... 10
- **Data** ........................................................................................................................................ 11
- **Managing Non-sightings** ........................................................................................................... 13
- **Thesaurus preparation** .............................................................................................................. 13
- **Eco-parser structure** .................................................................................................................. 14
- **The XSLT stylesheet** .................................................................................................................. 14
- **Javascript** .................................................................................................................................. 15
- **Perl scripts** .................................................................................................................................. 15
- **Evaluation and Discussion** ......................................................................................................... 16
- **Conclusion and future work** ....................................................................................................... 18
- **References** ............................................................................................................................... 22
Introduction
This paper will explore the use of geo-parsing in conjunction with information extraction techniques to document spatiotemporal incidences of ecological significance. For simplicity, this process will be referred to as ‘eco-parsing’. This paper will argue that valuable information can be extracted from bodies of text (corpuses) which is otherwise unavailable. The challenge is to accurately extract information that is relevant from complex linguistic structures and rendering this as useful data, which can be subjected to analysis within geographical information systems. The nature of the spatiotemporal incidence – the target for eco-parsing – should influence what contextual information is relevant for extraction. To illustrate this concept, tiger sightings will be extracted from a corpus of 19th century English language travel diaries for Asia, however the principal can be applied much more broadly to an increasing number of texts which are being digitised and made available (Ross, 2012). This corpus of unrestricted text will have a great degree of internal heterogeneity with respect to structure, syntax and information quality. Extraction of ecological information from historical texts with eco-parsing could provide additional data for ecological modellers that would not otherwise be discovered due to the number of texts involved being too great for manual parsing. The particular use case which will be examined in this paper is the identification of historical tiger occurrences. This data could add to models of total tiger range (Kitchener and Yamaguchi, 2010) and inform historical population baseline estimates for conservation purposes (McClenachan et al., 2012).

Research questions
- To what extent can geo-parsing and text mining techniques be utilised together to automate the extraction of spatiotemporal incidence information from unrestricted texts?

- How correct and efficient is the extraction of relevant contextual information for the target of eco-parsing?

- Within the chosen corpus of historical texts, was data richness or data quality the greater hindrance for eco-parsing for tiger sightings?
Applications of text parsing

Text mining and Natural Language Processing techniques have been put to great effect in the field of biomedicine, where the pace and scale of global research far exceeds any one individual’s ability maintain a complete overview of the subject (Huang and Lu, 2016). Recent advances in data mining techniques have proven promising for the development of precision medicine, where recognising the unique interconnections between specific genes, diseases and drugs is essential (Gonzalez et al., 2016). These new techniques allow vast quantities of natural language, unrestricted text from journals and digital libraries to be parsed for potential relationships that could not otherwise be achieved through manual means (Gonzalez et al., 2016). The biomedical field has also applied text mining techniques to literature derived from traditional Chinese medical texts as a method of knowledge discovery (Fang et al., 2008).

With the continued expansion of digital libraries that use Optical Character Recognition (OCR) scanning on historical texts, researchers are more able to take a macroscopic view of history and conduct text mining on a bulk scale (Graham et al., 2016). OCR scanned historical texts are significantly more prone to error than biomedical journals (Rupp et al., 2014), as well as, more heterogeneous in structure, language and style (Schneider et al., 2015). Therefore, parsing of historical texts has tended to be focused around sources with more consistent internal structures and styles. For example, Efremova et al. (2015) investigated Notary Acts as a source for identifying family relationships. Hinrichs et al. (2015) were able to visualise British and Canadian trade in the 19th century by analysing four major collections of trade documents. Furthermore, Gregory et al. (2016) were able to map major outbreaks of disease in 19th century England and Wales through parsing of newspaper archives. Spatial Humanities, a synthesis of classical humanities with GIS and computer linguistics, may still be in its infancy but it is also a rapidly developing research field (Gregory et al., 2015). This research project falls under the Spatial Humanities umbrella, while also incorporating some aspects of ecological science and biogeography.

Natural history researchers were earlier adopters of geo-referencing techniques to interpolate textual descriptions into coordinate systems (Beaman and Conn, 2003, Guralnick et al., 2006). However, these techniques were designed primarily to geo-parse short-form, detailed museum descriptions from the 20th century that were created deliberately for geographical reference. Beaman and Conn (2003) employ a highly autonomous process to parse herbarium archives compiled by botanists. Guralnick et al. (2006) developed their “BioGeomancer” as a tool for georeferencing batches of specimen record. Both of these tools operate under the assumption that their input data are verified specimens with intentional, if perhaps vague, locality descriptions. The main objective of these tools is data transformation; to apply coordinates to previously known and accessible data. By contrast, the eco-parsing process involves an element of data discovery through the parsing of non-specialist source texts, wherein, the ecological target might be referenced in numerous irrelevant contexts or have only an incidental geographic description.

The main challenge to geo-parsing and, by extension, eco-parsing is that the natural language from manuscripts is not machine-readable. Natural language is not sufficiently standardised in terminology or structure such that a machine can interpret it with certainty – as is required for formal computing languages (Lappin and Fox, 2015). The goal of geo-parsing is to transform a piece of natural language into a standardised geographic reference to increase its value and utility as spatial data (Alex et al., 2015). Eco-parsing is a process of extracting and standardising spatiotemporal ecological data from natural language sources. However, in order to target the correct spatiotemporal ecological incident for eco-parsing a formally structured thesaurus of search terms is required. Therefore, it is both thematically appropriate and practical that eco-parsing should
be integrated with wider movements to standardise data and, in particular, with “The Semantic Web” effort.

**The Semantic Web**

The Semantic Web provides a framework to support data sharing across software platforms and discipline boundaries by promoting a common set of languages and data standards (Burgos, 2011). These standards have been adopted by a number of fields with broad relevance to eco-parsing. Ordnance Survey utilise Semantic Web ontologies internally to streamline the generation of subject-specific maps (Goodwin, 2005). Semantic Web technologies provide a foundation for knowledge management and digital library systems with improved querying capabilities for sourcing desired texts (Anwarul Islam and Ikeda, 2014). Additionally, progress has been made within the field of biogeography to improve data access through the development of ontology-based biogeographical databases, underlain by Semantic Web standards (Henderson et al., 2007). Therefore, in the interest of promoting data integration between the overlapping disciplines that influence eco-parsing, several prescribed Semantic Web standards have been incorporated into the eco-parsing tool.

Four Semantic Web technologies are utilised by the eco-parser; XML, XSLT, RDF and SKOS. Extensible Markup Language (XML) is a syntax that employs elements to structure content within a document to enable both human and machine-readability (Kashyap et al., 2008). Extensible Stylesheet Language Transformations (XSLT) allows for the conversion of XML data into HTML and plain text for web page visualisation (Kashyap et al., 2008). The Resource Description Framework (RDF) provides a general methodology for information modelling and concept description within web resources (Burgos, 2011). Simple Knowledge Organization System (SKOS) builds upon RDF as a specific structure for controlled vocabulary content, including taxonomies and thesauri (Burgos, 2011). In addition to Semantic Web standards, geoJSON – based on Javascript Object Notation – provides an open standard for representing simple geographical features and their associated aspatial data (Butler et al., 2008).

**The potential the ecological value of historical texts**

Eco-parsing applied to historical texts could unlock new ecological data that would otherwise go unutilised. Figure 1 illustrates the range of relevant information that could potentially be derived from the textual description of a wildlife sighting.

*Figure 1: Potential information in a wildlife sighting*
The source texts for this research pre-date modern coordinate systems and even though “The Great Trigonometrical Survey” of India was on-going from 1802 to 1871, it is very unlikely that a personal diary would utilise any formal geographic reference system (Kennedy, 2007). Therefore, the available location data will be limited to place names and written descriptions. The temporal resolution of a sighting is depended upon the occurrences of dates within the source text; otherwise, a best estimate may be derivable from the date of publication. Information available on the target animal and context of the sighting will inevitably vary greatly with the richness of textual descriptions. Furthermore, uncertainty regarding the measurements and details of a sighting may not be acknowledged by the author at all, which reduces the reliability and resolution of data extracted.

Figure 2 is an excerpt from the 1877 travel diary of Henry M. Field, entitled “From Egypt to Japan” (Field, 1878). Field provides a description, as a secondary source, of a local man’s encounter with a Bengal tiger.

From the excerpt, it can be deduced that the sighting occurred at a “ravine by the roadside”, “a few rods in front of us”, “within five miles” of the man’s homestead. A place name – “Rajpore” – is given; however, it is implied to be the point of departure, not the destination. The necessary context to resolve this sighting is found several paragraphs earlier in the text, whereby it is established that the author was hosted at a Missionary Compound in “Dehra Doon”, managed by a Mr. Woodside and a Mr. Herron. Cross-referencing this information with entries from “History of Christian Missions: North India Perspective” (Sharma, 2005) reveals that the place names are historical spellings for the present day settlements of Rajpur and Dehradun, respectively.
The distance between these two settlements is approximately 6 miles of road (Figure 3). Furthermore, leopard sightings can be identified within the historic extent of Dehradun – perhaps a two-mile radius. Field’s account is an excellent example of both the potential spatial resolution available in historical sightings and the challenges faced when attempting to resolve them.
Eco-parser tool functionality

The eco-parser tool has been designed as an interface to discover, extract and visualise spatiotemporal incidences of ecological significance within historical texts. Firstly, a source text is filtered against a thesaurus of relevant search terms for the ecological target of choice. Subsequently, a series of rules within the eco-parser attempt to classify the context of each identified target term. The filtered source text is then visualised within a web browser, allowing the user to manually connect classified target ecological incidences to their location by pressing the available buttons (Figure 4). It can be seen in Figure 4 that the eco-parser classification rules successfully identified that the second occurrence of “tiger” in the page actually refers to the person “tiger hunter” rather than an animal.

Figure 4: Eco-parser button interface

The data connection process is simple. Firstly, a potential sighting would be chosen, then its location. On selection of a potential sighting, a prompt will appear to allow for editing of the prescribed classification. For example, if the identification of the “tiger hunter” as a person were to have been erroneous, the misclassification could be corrected and the incident properly recognised as an animal. Coupling of sightings and locations is possible both within and across page boundaries. In the case of unrecognised place names or ambiguous descriptions, a longer textual description may instead be selected by highlighting text with the cursor and submitting the excerpt via button press. Table 1 summarises the data extracted by the eco-parser and whether it is editable.
Table 1: Data extracted by eco-parser

<table>
<thead>
<tr>
<th>Data type</th>
<th>Example</th>
<th>User oversight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sighting page</td>
<td>356</td>
<td>Not editable, confirm or reject at sighting selection</td>
</tr>
<tr>
<td>Target phrase</td>
<td>tiger</td>
<td></td>
</tr>
<tr>
<td>Classification</td>
<td>“Wild tiger sighting”</td>
<td>Editable prompt upon selection of sighting</td>
</tr>
<tr>
<td>Latitude</td>
<td>22.000</td>
<td>Not editable, confirm or reject as location</td>
</tr>
<tr>
<td>Longitude</td>
<td>79.000</td>
<td></td>
</tr>
<tr>
<td>Location name</td>
<td>India</td>
<td></td>
</tr>
<tr>
<td>Location page</td>
<td>354</td>
<td></td>
</tr>
<tr>
<td>Optional location description (to be re-geoparsed etc.)</td>
<td>“A day’s ride from the Missionary Compound”</td>
<td>Selected via cursor highlighting and button press</td>
</tr>
</tbody>
</table>

Upon completion of manual selection, the sightings and their location data can be forward for geo-visualisation as geoJSON points on the web mapping service “geojson.io” – where they can be further edited and annotated by the user if required (Mandl et al., 2014)(Figure 5).

Figure 5: geojson.io visualisation

Data

Example historical manuscripts were sourced from the British Library’s “Microsoft Collection” – a digital collection of over 49,000 articles of 17th to 19th century English language literature. From this wider collection, 19th century works were prioritised since the 19th represents Britain’s “Imperial Century” – an era when English-speaking powers had the greatest overlap (Raj, 2007) with the habitat zones of tigers (Kitchener and Yamaguchi, 2010). In addition, 19th century works are better suited to parsing as their grammar and syntax are more similar to that of Present Day English.
(Schneider et al., 2015). Earlier 17th and 18th century works show greater variation in these rules, as well as, inconsistent spelling and the capitalisation of all nouns (Schneider, 2001, Archer et al., 2003). The capitalisation of all nouns is particularly inconvenient for Named Entity Recognition – the identification of proper nouns – that is a core component of most geo-parsing techniques (Mikheev et al., 1999, Gregory et al., 2015). Due to the scale of data available, a training corpus for the eco-parser was defined. Texts with a significant number of occurrences of both the word “tiger” and place names within India were identified to be used as training sources.

The entire “Microsoft Collection” was georeferenced in advance using the Edinburgh Geoparser, courtesy of Dr Claire Grover (Grover et al., 2010). The geo-parser output format of is XML; employing its tree structure to model the hierarchy between page, paragraph, sentence and word (Figure 6).

![Figure 6: Simplified XML book data tree structure](image)

Location data are encoded as attributes on each identified place name, in the form of latitude and longitude. Additionally, some basic textual analysis was conducted during the geo-parsing process and therefore each word has been labelled according to its function within the sentence – such as, noun, adjective or verb. The XML tree structure of data allows for the rules to be developed within the eco-parser tool that filter data and attempt to differentiate between contexts. For example, pages that do not reference the target ecological incident or contain location data are excluded from
being displayed to the user. The eco-parser is also able to extract attribute data from the Page level of the XML tree to provide page references for each sighting and location.

**Managing Non-sightings**

Not every occurrence of a target animal within a text will be an actual sighting of that animal in the wild. In addition to sightings in the wild, four other broad categories were deduced for scenarios in which the target animal is referenced. These categories will be discussed further as “Domesticated”, “Product”, “People” and “Fictional”.

‘Domesticated’ refers to cases wherein the target animal is captive or tame. Simple examples of how this category might be expressed within a text would be; “caged tiger” or “the tiger was caged”.

‘Product’ encompasses incidences of the target animal being deceased and where reference is made only to its body parts. A “tiger skin rug” and “tiger bones” are examples of this category.

‘People’ denotes situations in which the animal term is instead related to a person, as part of an occupational title or an activity that they are undertaking. Examples of this would be “tiger hunter” and “tiger stalking”.

‘Fictional’ attempts to capture any non-factual occurrences of the target animal, including poetry, metaphor and simile. An example for a simile would be; “he was as fierce as a tiger”. A metaphor might be; “he has the eyes of a tiger”.

The above categories are broad, complex and their identification quickly becomes a problem of natural language processing in computer linguistics and machine learning. Furthermore, considering the variable quality of sources and the potential difficulties of extracting valid data, priority for the eco-parser should be focused on maximising the number of inclusion of prospective tiger sightings and minimising any accidental exclusions. Therefore, automated categorisation of sightings cannot be solely reliable upon and manual oversight for data selection will still be required. Instead, automated categorisation has been designed as prescriptive only – not authoritative – and, consequently, all proposed classifications can be manually overridden.

**Thesaurus preparation**

A thesaurus of search terms for the target spatiotemporal incident is needed to run the eco-parser. In order to demonstrate best practice for thesaurus building and ensure usability to persons unfamiliar with SKOS data standards, a free, open-source, web-based SKOS editor was used to create the tiger thesaurus (Conway et al., 2016). However, due to the hierarchical structure of SKOS files, not all terms input into a thesaurus are stored as text strings within XML elements. Therefore, an initial XSLT stylesheet was applied to extract all relevant terms as strings and repackage them into a simplified list of XML elements. A second XSLT stylesheet was then implemented to convert the extracted terms to lowercase. All stylesheets for this research project were applied using the Edinburgh Geoparser compiler (Grover et al., 2010) (Figure 7).
Eco-parser structure

The eco-parser utilises three core technologies in its functioning; Extensible Stylesheet Language Transformations (XSLT), Javascript and Perl. Figure 8 summarises the flow of data through the core components of the eco-parser. The XSLT stylesheet generates the HTML visualisation of a source text, with which the user interacts. Javascript is utilised to collect and pass parameters from the stylesheet rules on to the Perl scripts. These Perl scripts then package the given parameters into output files for geo-visualisation or reference.

The XSLT stylesheet

The Ecoparser is initiated by applying an XSLT stylesheet, along with the XML thesaurus of search terms, to a target geoparsed, XML book. The Ecoparser stylesheet filters out XML annotation and irrelevant text. Rules encoded within the stylesheet provide basic, automated decision-making functionality, which highlights text areas of interest and prescribes an initial classification schema any references to the target animal. The automated decision-making can be influenced by editing XML files for the four broad soft-exclusion categories; Domesticated, Fictional, Product and People. The classification method approximates the context of use for each target animal term by word-
matching their near-neighbours against exclusion lists for the four non-sighting categories. If a target animal term occurs in close proximity to – generally first or second neighbour and before or after – a phrase from an exclusion list, then the animal term is flagged with a classification other than “Wild animal sighting”. No data is output until the user has manually resolved or confirmed the prescribed classification.

Javascript
The Ecoparser stylesheet also connects to an imported Javascript file, whose functions extract data from XML elements within the underlying book file – based on user decisions input through the browser visualisation (Figure 9). The Javascript creates a series of global variables for extracted data as the user manually confirms a sighting and couples it with a location. Table 1 outlines the data collected and how a user may interact with it. Each confirmed couple of sighting and location is then sent on via GET request to a Perl script. Sighting data can only be sent forward once it has been coupled with a location, or given an optional location description.

Figure 9: Script interactions

Perl scripts
An initial Perl script is activated as the browser visualisation is opened – by utilising an embedded Javascript ‘onload()’ within the Ecoparser stylesheet. This script creates a new .geojson file with the opening segment of a “FeatureCollection” – a data standard for storing geometry objects (Butler et al., 2008). The main script receives data extracted by the Javascript as parameters, which it then uses to populate the structure of a geoJSON point object and finally outputs the point object into the same .geojson file as above. A third script is activated upon pressing the end button at the bottom of the browser visualisation to enclose the geojson file and thus create a valid “FeatureCollection” for exporting. The final Perl script is activated by a Javascript prompt window generated upon pressing
the end button. This script packages the geoJSON file correctly so that it can be open in geojson.io for visualisation.

Evaluation and Discussion

A prototype eco-parsing tool was successfully developed to parse an underutilised resource of historical texts. The tool can be applied to greatly reduce the volume of irrelevant information and promote desired targets for data extraction to the user. Figure 10 is an example of tiger sighting data extracted by the eco-parser and converted into geoJSON format for visualisation in geojson.io.

*Figure 10: Eco-parser geovisualisation through geojson.io*
Several non-expert users tested the eco-parser tool and gave candid feedback. Positive comments focused on the convenience of the tool compared to manual reading and mapping of points (Table 2). Feedback for improvements centred on user interface, which was not a major design consideration for the eco-parser prototype. It was also suggested that a reference file for context and text excerpts could be created in parallel to the geoJSON file with unique corresponding ID numbers to later link them as attributes in a formal GIS software. This would be a solution to the geoJSON structure not being well suited for storing and displaying large strings of text.

Table 2: Positive feedback from eco-parser tool tests

<table>
<thead>
<tr>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>“It’s really useful to be able to add a brief context through the classification prompt.”</td>
</tr>
<tr>
<td>“Being able to manually link a subject of interest to a geographic location then visualise it is powerful.”</td>
</tr>
<tr>
<td>“The automated classification feature is useful at a glance to ignore things that are less likely to be sightings and come back to review them later.”</td>
</tr>
<tr>
<td>“Having the page numbers for each sighting and location makes this a useful tool to quickly note down ecological targets with preliminary locations. You could then go back and read the book fully or apply other textual analysis tools to get a more accurate location.”</td>
</tr>
<tr>
<td>“It will cut out a lot of research time. There is no way I could or would want to read ten, 300+ page old travel diaries to find maybe 20 moderately useful data points between them.”</td>
</tr>
<tr>
<td>“Switching between plain text and XML code to manually couple each sighting and location would be very laborious and prone to error. Having a user interface that is designed to simplify this process is useful.”</td>
</tr>
</tbody>
</table>

Data quality and consistency were quickly identified as major challenges to the efficacy of an eco-parsing tool. The quality of data extracted is, necessarily, determined by the quality of the sources input. Frequently encountered data quality issues included; Optical Character Recognition errors and missing, partial or inferred titles, authors and dates within the metadata. However, in order to delineate the scope of design for the eco-parsing tool developed in this project, it was determined that data quality issues should be addressed independently by the user. For the purposes of developing and demonstrating the functionality of this eco-parsing tool, only texts with sufficient data quality and a high occurrence of references to tigers were selected.

The quality of tiger sighting descriptions varied greatly between sources. While Figure 1 outlined the range of information that would be extracted from an ideal sighting description, very few excerpts contained detail approaching this level. Therefore, the aims of the project were realigned to capture a consistent, base-line of data with an automated soft classification scheme and greater user oversight as seen in Table 1. This decision reaffirmed the primary purpose of the eco-parser as a geo-referencing and visualisation tool for spatiotemporal incidents of ecological significance – as opposed to specialist data mining and textual analysis packages.

The quality of eco-parsing data is also directly related to the methodology and resolution of the geo-parsing applied upon source texts. If the geo-parser does not recognise a place name then no latitude and longitude are assigned to that place name. Therefore, the eco-parser will not offer the
place name as a location for resolving an animal sighting. An alternative method was devised to mitigate this issue, which allows the user to manually highlight a section of text with their cursor and submit that text as a location description along with the standard sighting details. This enables the user to either record a place name not identified by the geo-parsing progress that could potentially be geo-parsed again, or capture a longer form textual description as the location.

Eco-parsing of historical texts, such as travel diaries, is a challenge of extracting useful spatiotemporal data of ecological significance from sources that were not created specifically for this purpose. Additionally, the author may have no expertise in the region, navigation or ecology. In this regard, eco-parsed historical data should be treated with greater care and scepticism than citizen science, since there would not have been subject-specialist-informed framework with which to record animal sightings (Bonney et al., 2009).

**Conclusion and future work**

This paper has explored the potential of combing conventional geo-parsing some basic textual analysis techniques in order to extract information on spatiotemporal incidences of ecological significance. This process could be referred to as “eco-parsing” in shorthand. A prototype eco-parsing tool was developed to demonstrate this concept by targeting tiger sightings in a corpus of 19th century manuscripts related to India, which were sourced from the British Library’s “Microsoft Collection”. The eco-parsing tool was initially intended to have a high degree of automation and extract a wide range of contextual information for a target sighting. However, upon assessing the heterogeneity of text structures and the variability of data quality, it was realised that significant automation would require more application of computer linguistics technology and that this process would likely result in more accidental exclusions of meaningful data. Therefore, to minimise potential data loss, the eco-parser was redesigned as a semi-automated classification and visualisation tool to present the user with an overview of all the identified potential target animal sightings within a source text. Thus, more complex text excerpts can be manually resolved by the user, where the automation might otherwise have failed. The result is that the eco-parser can be utilised to more reliably capture a base-line of data from both limited and overly complex sighting descriptions.

The implemented automatic classification system was based upon four broad exclusion categories that could be applied where a target animal phrase was referenced in a manuscript but did not describe a valid wildlife sighting. The exclusion categories encompass; domesticated or captured animal, animal-related product, animal-related profession or human activity and fiction – including simile and metaphor. A classification system can be influenced by the user through editing XML exclusion lists and any prescribed classification can be manually edited. Semantic Web standards and geoJSON spatial data encoding were utilised within the eco-parser to promote compatibility across the range disciplines that could contribute to the future development eco-parsing.

In answer to the first-posed research question, the extent to which automation can be successfully applied to eco-parsing will be limited by the generic challenges faced in all forms of parsing; the structure and consistency of the data source and the tolerance for accidental data exclusion. Therefore, the degree of automation to utilise within an eco-parsing project should be decided on a case-by-case basis, considering the scale of the corpus and interest in marginal or difficult to resolve incidences.
Both data quality and data richness were found to be a hindrance to the eco-parsing process. Data quality was the more overt challenge as it limited the degree of automation and forced a more minimalist, base-line approach to data extraction. However, poor data richness also necessitates a more careful and manual approach to eco-parsing as false exclusions are costlier where data is sparse. Having fewer examples of animal sightings also limits the opportunity for training an eco-parser on real data.

Due to the cross-disciplinary nature of eco-parsing, future development of the tool would depend on the needs of the target user groups and software gaps. Generally, further development could focus on; data source management, improving automated categorisation, improving the web service, broadening the scope of targetable spatiotemporal incidents, increasing contextual information extraction, incorporating basic spatial analysis techniques or improving geo-visualisation.

Before engaging in other areas of development for eco-parsing, however, priority should be placed on the identification and classification of uncertainty within extracted data. Uncertainty encompasses the veracity of sightings and the ambiguity and vagueness of their spatial and temporal descriptions (Fisher, 1999). As the current Eco-parser requires manual coupling for sightings and locations, the responsibility for accessing authenticity devolved to the user – with an assumed degree of expertise in their field. However, in the interest of acknowledging uncertainty and promoting the responsible use of spatial data, the user could be required to include a metric of confidence in the veracity of each sighting before any data may be extracted. In addition, an automated comparison of all output geometries against the expected distribution for each target species could offer a ranking schema to identify marginal cases of interest and exclude obvious errors. Implementation of this basic spatial analysis could be delivered through a customised web mapping service, with spatial query functionality, which imports distribution shapefiles for target species from online resources – such as the IUCN Red List (IUCN, 2001) (Figure 11). The eco-parser web mapping service could also attempt to import historical maps from the same region and closest time period.
Incorporating a method to automatically target a source text’s metadata would allow for the extraction of book titles and dates along with the file name and page references. Any missing, partial or inferred titles, authors and dates could also be flagged in an uncertainty ranking schema for extracted data. In addition, the tool could be developed to attempt to exclude works of fiction by cross-referencing titles with online catalogues and a list of terms to reject – such as “poetry”.

The automated classification methodology could be expanded to include rules that attempt to identify more complex sentence structures and metaphors. Furthermore, a rule could built into the XSLT stylesheet whereby a target animal sentence that is not classified by one of the more simplistic stylesheet rules is then automatically forwarded to a Perl script that utilises formal textual analysis modules to help resolve the classification. Work could also be done to incorporate the hierarchy of target phrases imported from the SKOS thesaurus. Phrases that are recognised as being more broad
potential terms for the target animal, such as “beast”, could be classified as less likely sightings. Whereas terms that are much narrower, such as “tiger”, would be classified as being more likely to be a valid sighting.

Increasing the amount contextual information that is automatically targeted for extraction could be explored with the use of the attribute labels encoded within the XML element of each word. For example, targeting the closest verb in the sentence containing a target animal reference might yield the activity of said target animal. Perl textual analysis modules could also be applied to this challenge in a similar manner to the classification method. In addition, there may also be non-textual information in the form of diagrams or sketches associated with an animal sighting. Pictures from the original manuscripts scanned by the British Library are available with page references on the image-hosting website Flickr. An additional script could be created to trawl the Flickr British Library collection for images in close page proximity to animal sightings.

For the purpose of demonstrating the potential value of eco-parsing in this research project, the eco-parsing tool was developed to most effectively target tiger sightings. The automated classification schema was designed and trained around example tiger sighting excerpts. This classification schema in its current form would not be appropriate for plant species and would also not apply as reliably to other animal species. Work could be done to broaden the compatibility revaluating the four non-sighting categories, perhaps adding additional categories for plants, and training them further on other text excerpts.

Since the extracted spatial data conforms to a recognised standard – geoJSON – it could be transferred to a formal GIS software package for further spatial analysis. However, there would be a great variety of uncertainty within the extracted data to consider. Geo-visualisation would be an effective method of communicating the spatial uncertainty of historical animal sightings. Wieczorek et al. (2004), Guo et al. (2008) and (Erp et al., 2015) offer methodologies for converting textual descriptions into spatial data that also display the associated uncertainty for each derived locality. Wieczorek et al. (2004) use a point-radius method, which denotes the most fitting location with a radius of relative uncertainty surrounding it. Guo et al. (2008) employ a probabilistic approach to account for uncertainty in both the reference location and its spatial relationship to the derived locality – producing an appropriately shaped field of probability. While, Erp et al. (2015) incorporate a confidence metric based upon the complexity of processes used to derive spatial data.

In conclusion, eco-parsing of historical manuscripts offers a new and potentially valuable source of conservation data. Eco-parsing is limited by the challenges of extracting useful data from sources that were not created specifically to convey ecological data. Consequently, there is significant uncertainty within eco-parsed data and an elevated potential for data misunderstanding and misuse. Therefore, future research should prioritise the modelling and visualisation of the uncertainty associated with the eco-parsing process.
References


FIELD, H. M. 1878. From Egypt to Japan. JSTOR.


Table of Contents for Technical report

Introduction ........................................................................................................................................... 26
Initial Experiments ............................................................................................................................... 26
Final Computing Environment ............................................................................................................ 27
Edinburgh Geoparser ........................................................................................................................... 28
Thesaurus preparation ........................................................................................................................... 29
Eco-parser structure ............................................................................................................................. 30
XSLT stylesheet .................................................................................................................................... 30
Javascript ................................................................................................................................................ 33
Perl ...................................................................................................................................................... 35
Eco-parser user interface ...................................................................................................................... 36
References ............................................................................................................................................ 40

Table of figures

Figure 1: Potential information in a wildlife sighting................................................................. 7
Figure 2: Travel diary excerpt of tiger sighting (Field, 1878) ......................................................... 8
Figure 3: Mock-up geovisualisation of wildlife sightings on an 1879 map (adapted from: PAHAR (2013)) ........................................................................................................................................... 9
Figure 4: Eco-parser button interface ............................................................................................ 10
Figure 5: geojson.io visualisation ..................................................................................................... 11
Figure 6: Simplified XML book data tree structure ....................................................................... 12
Figure 7: Thesaurus preparation ........................................................................................................ 14
Figure 8: Eco-parser tool data flow .................................................................................................... 14
Figure 9: Script interactions .............................................................................................................. 15
Figure 10: Eco-parser geovisualisation through geojson.io ......................................................... 16
Figure 11: Mock-up of historical tiger sighting against modern range (IUCN, 2001) .................. 20
Figure 12: Early test script .................................................................................................................. 26
Figure 13: Early test script output ..................................................................................................... 27
Figure 14: String extraction template ............................................................................................... 29
Figure 15: Thesaurus processing ........................................................................................................ 29
Figure 16: Eco-parser tool data flow ............................................................................................... 30
Figure 17: Template to display pages that contain a phrase comprising of two words ............. 31
Figure 18: Simplified XML book data tree structure ..................................................................... 31
Figure 19: Template contains multiple tests for the neighbouring words around a tiger phrase .... 32
Figure 20: Passing XML element attribute data to Javascript function ......................................... 33
Figure 21: Javascript accepting XML data as arguments ................................................................. 33
Figure 22: Javascript get request sending parameters to Perl script .......................................... 34
Table of tables

Table 1: Data extracted by eco-parser .................................................................................. 11
Table 2: Positive feedback from eco-parser tool tests ...................................................... 17
Introduction
This document will outline areas of code and techniques that are essential to the functioning of the eco-parser tool.

The challenge of an eco-parser is to accurately extract information that is relevant from complex linguistic structures and rendering this as useful data, which can be subjected to analysis within geographical information systems. The nature of the spatiotemporal incidence – the target for eco-parsing – should influence what contextual information is relevant for extraction. The contextual information informs the automatic classification to direct the user to potentially relevant additional information which allows manual selection for improved resolution and accuracy.

A system was implemented which combines an existing geoparsing tool, with a specific ontology for the example species (tigers) together with a mapping tool which displays the results extracted in industry-standard GeoJSON format, which can easily be imported into geographical information systems.

Initial Experiments
Initial work used a series of Perl programs to examine an example plain-text corpus from Project Gutenberg, which represents a travel diary “From Egypt to Japan” by Henry M. Field (1878) (Figure 12). The target word occurrences were returned together with some limited contextual information (Figure 13).

Figure 12: Early test script

```perl
#!/usr/bin/perl
use DBI;
use CGI qw(:standard);
use CGI::Carp qw(fatalsToBrowser);

print "Content-Type: text/html\n\n";
print qq(<HTML>
);
print qq(<HEAD>
);
print qq(<TITLE>Example Search</TITLE>
);
print qq(</HEAD>
);
print qq(<BODY>
);

$file = $ARGV[0];
$word = $ARGV[1];
# rather simplistic word search
$word = " ".$word." ";

open (FILE, $file) or die "Couldn't open file $!";
$string = lc(join("", <FILE>));
close FILE;

$pos=1;
$occ=0;
```
while ($pos != 0) {
    $pos = index($string,$word,$pos);
    print "Occurrence $occ: ".substr($string,$pos-30,60)."\n";
    $occ = $occ + 1;
    $pos = $pos + 1;
}
print "\n";

<HTML>
<HEAD>
<TITLE>Example Search</TITLE>
</HEAD>
<BODY>
Occurrence 0: describe the particulars of a tiger hunt--how the game of a
Occurrence 1: l at last a magnificent bengal tiger sprang into view, and
Occurrence 2: tertain him withthe grandest tiger hunt ever known in indi
Occurrence 3: h a lady, a magnificent bengal tiger came up out of that
Occurrence 4: ne the less romantic because a tiger had once carried off a
Occurrence 5: riding this morning was an old tiger hunter, that had often
Occurrence 6: then in terror because of a tiger who had lately come abo
Occurrence 7: nly let the carcass alone, the tiger always comes back, and
Occurrence 8: en just at break of day a huge tiger walked out of a wood,
Occurrence 9: same as the right of a bengal tiger to his jungle--a right
Occurrence 10: n mistake, which shows that a tiger is near. doubtless he w
Occurrence 11: lmost immediately fatal. these tiger skins or elephant tusks
Occurrence 12: he home of wild beasts--of the tiger and the rhinoceros. wil
Occurrence 13: the leafy solitudes, but the tiger and the rhinoceros come

Final Computing Environment
The Edinburgh Geoparser is run from the command line within the burn.geos.ed.ac.uk compute server, which then generates an HTML file accessible via the xweb.geos.ed.ac.uk web server. This is displayed on a web browser, allowing user interaction to semi-automatically identify tiger locations which, when confirmed, are relayed to a temporary geoJSON file on the xweb server such that these can be rendered using the GeoJSON.IO rendering engine enabling the user to interact with a openlayers-style map. Data can be extracted as geoJSON for further analysis as required.
Edinburgh Geoparser

The Edinburgh Geoparser is a language processing tool designed to detect placename references in English text and ground them against an authoritative gazetteer so that they can be plotted on a map (Alex et al., 2015). The two main processes involved are entity recognition, to find the placename mentions and categorise them as such, followed by a ranking process that selects the likeliest location for each place from what may be a long list of candidates (Grover et al., 2010).

The geoparser was developed by Claire Grover and Richard Tobin, of the Language Technology Group in the School of Informatics at the University of Edinburgh. The Geoparser has formed the basis for several projects and the UNLOCK service provided by EDINA, and is acknowledged as one of the best tools of its type available.

Like many linguistic tools of this kind, the geoparser software is designed to work in a “pipeline”, where the output of one process forms the input for the next. This construction gives flexibility and makes it easier to customise the components (Grover et al., 2010).

The geoparser is configured to work with a number of different gazetteers, providing a good level of placename coverage across the globe. These gazetteers include geonames (Geonames, 2016), natural earth (Natural Earth, 2016) and Pleiades+ (Stoa Consortium, 2008).
Thesaurus preparation

Even though the thesaurus of tiger search terms was constructed in a Semantic Web standard Simple Knowledge Organization System (SKOS) web tool, not all search terms were accessible as strings within XML elements (Conway et al., 2016). Therefore, the SKOS tiger thesaurus had to be reconstructed with all relevant strings accessible. This was achieved by applying an extraction XSLT stylesheets using the Edinburgh Geoparser compiler (Grover et al., 2010) (Figure 14) (Figure 15).

**Figure 14: String extraction template**

```xml
<xsl:template match="owl:NamedIndividual">
  <xsl:apply-templates select="skos:altLabel|skos:prefLabel"/>
</xsl:template>

<xsl:template match="skos:altLabel|skos:prefLabel">
  <Group>
    <xsl:text>&#10;</xsl:text>
    <xsl:text>&#09;</xsl:text>
    <Name1><xsl:value-of select="node()" /></Name1>
    <xsl:text>&#10;</xsl:text>
    <xsl:text>&#09;</xsl:text>
    <Name2><xsl:value-of select="substring-after(../@rdf:about, '#')" /></Name2>
    <xsl:text>&#10;</xsl:text>
    <xsl:text>&#09;</xsl:text>
    <Name3><xsl:value-of select="substring-after(../skos:broader/@rdf:resource, '#')" /></Name3>
  </Group>
</xsl:template>
```

**Figure 15: Thesaurus processing**

![Thesaurus processing diagram](image-url)
Eco-parser structure
The eco-parser utilises three key technologies to extract data from Extensible Markup Language (XML) encoded source texts. These are; Extensible Stylesheet Language Transformations (XSLT) stylesheets, Javascript and Perl (Figure 16).

Figure 16: Eco-parser tool data flow

XSLT stylesheet
An XSLT stylesheet is applied to the source text by the Edinburgh Geoparser compiler. The Edinburgh Geoparser must be run from a Linux terminal with the following command structure:

```
Structure:
cat xml book file | Edinburgh Geoparser compiler ≤ xslt stylesheet ≤ output html document
Example:
cat 000164914_0_1-938pgs__980883.xml | /home/s1581683/dissertation/data/geoparser-v1.1/bin/sys-i386-64/lxt ≤ fulltest.xsl ≤ newtest.html
```

The stylesheet applied templates to the XML tree structured book file so that only relevant text was displayed in the HTML page it generated. A template is only applied if the template match condition is met. However, the template match condition applies per individual string when matched against a document, such as a list of tiger phrases. Therefore, if a phrase contains more than one string it must be concatenated into one string to match correctly (Figure 17).
Templates can be applied to and access any level for the XML tree structure, including individual words (Figure 18). This allows for templates to be applied to one level, based on conditions met at another level. A simple example would be; highlight a sentence if it contains the word “tiger”. This allows for an individual word to have different template applied to it base on its context – its neighbouring words and other words in the page or paragraph.

Figure 18: Simplified XML book data tree structure
Using the conditions ‘choose’, ‘when’ and ‘otherwise’ and targeting either ‘follow-sibling’ or ‘preceding-sibling’ the same word could have a different template applied to it depending on its neighbouring words (Figure 19).

![Figure 19: Template contains multiple tests for the neighbouring words around a tiger phrase](image)

```xml
<xs:template match="w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerlist)//*]">
    <xs:choose>
        <!-- highlight tiger related persons -->
        <xs:when test="following-sibling::*[1][self::w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerperson)//*]]">
            <span class="red">
                <script src="sendlocs.js"></script>
                <button onclick="myFunction7('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}', '{node()}/@id', '{ancestor::node()}/@PHYSICAL_IMG_NR}')">Person?</button>
            </span>
        </xs:when>
        <xs:when test="following-sibling::*[2][self::w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigercaptive)//*]]">
            <span class="red">
                <script src="sendlocs.js"></script>
                <button onclick="myFunction10('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}', '{node()}/@id', '{ancestor::node()}/@PHYSICAL_IMG_NR}')">Simile?</button>
            </span>
        </xs:when>
        <!-- highlight tiger sighings -->
        <xs:otherwise>
            <span class="tiger">
                <script src="sendlocs.js"></script>
                <button onclick="myFunction3('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}', '{node()}/@id', '{ancestor::node()}/@PHYSICAL_IMG_NR}')">Sighting?</button>
            </span>
        </xs:otherwise>
    </xs:choose>
</xs:template>
```

Templates can also be used to access the elements and attributes of the XML document so that they can be passed as parameters to Javascript functions (Figure 20).
Figure 20: Passing XML element attribute data to Javascript function

```xml
<!-- mark locations as spans of class 'loc' - these will be highlighted in blue -->
<!-- also sends on lat/long to confirmation script that can trigger perl -->
<xsl:template match="enamex[@type='location']">
  <span class="loc">
    <xsl:apply-templates/>
  </span>
</xsl:template>
```

Javascript

Javascript was the intermediate language used to transfer data from XML to Perl to be formatted (Figure 21).

```javascript
// Function for location confirmation buttons (shows coordinates)
function myFunction4(name,latitude,longitude,gaz,page,word, context) {
    loc = name
    lat = latitude
    long = longitude
    locpage = page
    gazref = gaz
    coords = "Location: " + name + "\nLatitude: " + lat + "\nLongitude: " + long + "\nGazetteer reference: " + gaz + "\nPage: " + page + "\nWord ID: " + word;
    if (confirm("Sighting selected:\nSighting selected:\nProposed sighting location: \n\nConfirm this sighting and location then send to output?\n\n") == true) {
        // triggers confirmation alert
        alert(coords + "\n\nCoordinates sent.");
        // triggers perl sending function
        loadDoc4();
    }
}
```

A series of global variables were created to temporarily store extracted element and attribute data for sightings and locations. Selecting a new sighting or location would, therefore, overwrite the respective temporary data. So after each coupling of sighting and location the temporary data was sent via get request to Perl for writing to output (Figure 22).
Figure 22: Javascript get request sending parameters to Perl script

```javascript
// Function to send coordinates to perl script for geojson encoding
function loadDoc4() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML = this.responseText;
    }
  }
  xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/geojson.pl?fname=\"+encodeURIComponent(loc)\"&lat=\"+encodeURIComponent(lat)\"&long=\"+encodeURIComponent(long)\"&locpage=\"+encodeURIComponent(locpage)\"&gazref=\"+encodeURIComponent(gazref)\"&sipage=\"+encodeURIComponent(sipage)\"&clas=\"+encodeURIComponent(clas), true);
  xhttp.send();
}
```

Javascript functions activate Perl scripts in a variety of ways through the HTML Web interface (Figure 23).
Perl

Perl is used to create the geoJSON output file. The second script writes out the geoJSON format (Figure 24).

Figure 24: Perl script to write geoJSON structure

```perl
open(FILE, ">>$file") or die("Unable to open file $file : $!");
print FILE "\n"
print FILE "\n"
print FILE "\"type\": \"Feature\",\n"
print FILE "\"geometry\": \n"
print FILE "\"type\": \"Point\",\n"
print FILE " \"coordinates\": [".$long.", ".$lat."]\n"
print FILE "},\n"
print FILE "\"properties\": {\n"
print FILE "\"clas\": \".$clas.\",\n"
print FILE "\"sight_page\": \".$sipage.\",\n"
print FILE "\"name\": \".$name.\",\n"
print FILE "\"loc_page\": \".$locpage.\",\n"
print FILE "\"gazetteer_ref\": \".$gazref.\",\n"
print FILE "]},\n"
```

In order to complete the geoJSON file the “FeatureCollection” must be closed. This requires the removal of an additional comma that is added by the second script between each geoJSON point. The script shown in Figure 25 removes the last two characters from the geoJSON file before adding the brackets required to close the “FeatureCollection” and the file (Butler et al., 2008).

Figure 25: Ending the geoJSON file

```perl
my $ssize = -s $file;
open(FILE, "+<", $file) or die("Unable to open file $file : $!");
seek FILE, $ssize-2, 0;
print FILE "\n"
print FILE "\n"
print FILE "]\n"
close FILE;
```

The last Perl script packages the geoJSON file with an “Access-Control-Allow-Origin” header, which enables the geojson.io editor to receive and visualise it (Figure 26).
my $cgi = CGI->new;
my $file = "/web/s1581683/public_html/dynamic/parser_points.geojson";

local $/;

print "Access-Control-Allow-Origin: *\n"

open(FILE, $file) or die "Can't read file 'filename' [$!]\n";
my $document = <FILE>;
close (FILE);
print $document;

---

Eco-parser user interface

The user interface changes partly depending on the composition of the underlying XML book page. Figure 27 shows the interface for a page with no sightings, only a location. Figure 28 displays the interface for a page with multiple references to a potential tiger.
Upon selecting a sighting the user is prompted to accept a prescribed automatic classification or modify it to reflect their manual assessment of the potential sighting (Figure 29).

Figure 30 displays a brief pop-up alert, which confirms to the user that contextual information for their selected sighting has been temporarily stored.
Some of the context captured for a sighting is brought back up to remind the user of which sighting they are working with when they select a location (Figure 31). If the user cancels, the context for the last sighting selected will remain, so they may browse other locations without having to repeatedly re-select the same sighting over again.
Figure 32 show the geojson.io editor with points sent to it from the eco-parser. These point can be edited through the map interface via “drag and drop” or through editing their encoding. Other shapes and features can be drawn added by code into the geovisualiser as well. Most crucially for the user, geojson.io is intuitive and data can easily be exported from it.
References


FIELD, H. M. 1878. From Egypt to Japan. JSTOR.


Appendixes

Command line

<table>
<thead>
<tr>
<th>Structure:</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat xml book file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat 000164914_0_1-938pgs__980883.xml</td>
</tr>
</tbody>
</table>

XSLT Stylesheets

<table>
<thead>
<tr>
<th>Eco-parser XSLT stylesheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!DOCTYPE xsl:stylesheet [</td>
</tr>
<tr>
<td>&lt;!ENTITY nbsp &quot;&amp;\xa0;&quot; &gt;</td>
</tr>
</tbody>
</table>
| ]>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:param name="tigerlist" select="'tigertestfinal.xml'"/>
<xsl:param name="tigerproduct" select="'tigerproduct.xml'"/>
<xsl:param name="tigerperson" select="'tigerperson.xml'"/>
<xsl:param name="tigercaptive" select="'tigercaptive.xml'"/>

<xsl:template match="/">
<html>
<head>
<title>Geoparser Output</title>
<style>span.tiger {background:#04B431}</style>
<style>span.loc {background:#33ccff}</style>
<style>span.red {background:#FE2E2E}</style>
<b><xsl:text>Instructions:</xsl:text></b><br/>
<xsl:text>1. Choose a target tiger sighting with a "Sighting?" button.</xsl:text><br/>
<xsl:text>2. Select its location with a "Location" button</xsl:text><br/>
<xsl:text>OR</xsl:text><br/>
<xsl:text> manually highlight a location with your cursor and press any "Confirm manually highlighted location" button.</xsl:text><br/>
<xsl:text>3. Once finished confirming all sightings, press "Finish" button to complete your session and generate geojson file</xsl:text><br/>
<script src="sendlocs.js"></script>
</head>
<body onload="loadDoc1()">
<button onclick="loadDoc5()">Finish - generate geojson</button>
</body>
</html>
Page: possible tiger sighting.

Page: possible tiger product.

<!-- do display pages which contain a tiger word and a tiger product term -->
<esi:template match="Page[.//w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerlist)//*] and following-sibling::*[1][self::w[translate(concat(., ' ', following-sibling::w), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerproduct)//*]]]">
<p>
<script src="sendlocs.js"></script>
</p>
<hr/>
<button onclick="myFunction2()">Confirm manually highlighted location</button>
</esi:template>

<!-- highlight tiger words -->
<!-- rules to classify identified tiger terms -->
<xsl:template match="w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerlist)//*]">
  <xsl:choose>
    <!-- highlight tiger related persons -->
    <xsl:when test="following-sibling::*[1][self::w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerperson)//*]]">
      <span class="red">
        <xsl:apply-templates/>
        <script src="sendlocs.js"></script>
        <button onclick="myFunction7('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}','{node()/@id}', '{ancestor::node()[4]/@PHYSICAL_IMG_NR}')">Person?</button>
      </span>
    </xsl:when>
    <!-- highlight tiger product -->
    <xsl:when test="following-sibling::*[1][self::w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigerproduct)//*]]">
      <span class="red">
        <xsl:apply-templates/>
        <script src="sendlocs.js"></script>
        <button onclick="myFunction8('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}','{node()/@id}', '{ancestor::node()[4]/@PHYSICAL_IMG_NR}')">Product?</button>
      </span>
    </xsl:when>
    <!-- highlight tiger similes -->
    <xsl:when test="preceding-sibling::*[2][self::w[.='as'] and preceding-sibling::*[4][self::w[.='as']]]">
      <span class="red">
        <xsl:apply-templates/>
        <script src="sendlocs.js"></script>
        <button onclick="myFunction9('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}','{node()/@id}', '{ancestor::node()[4]/@PHYSICAL_IMG_NR}')">Simile?</button>
      </span>
    </xsl:when>
    <!-- highlight captive tiger -->
    <xsl:when test="preceding-sibling::*[1][self::w[translate(., 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz') = document($tigercaptive)//*]]">
      <span class="red">
        <xsl:apply-templates/>
        <script src="sendlocs.js"></script>
        <button onclick="myFunction10('{node()}','{preceding-sibling::w[3]}','{preceding-sibling::w[2]}', '{preceding-sibling::w[1]}','{following-sibling::w[1]}','{following-sibling::w[2]}', '{following-sibling::w[3]}','{node()/@id}', '{ancestor::node()[4]/@PHYSICAL_IMG_NR}')">Simile?</button>
      </span>
    </xsl:when>
  </xsl:choose>
</xsl:template>
<xsl:template match="enamex[@type='location']">
  <span class="loc">
    <xsl:apply-templates/>
  </span>
</xsl:template>

<!-- mark locations as spans of class 'loc' - these will be highlighted in blue -->
<!-- also sends on lat/long to confirmation script that can trigger perl -->
<xsl:template match="enamex[@type='location']">
  <span class="loc">
    <xsl:apply-templates/>
  </span>
</xsl:template>

</xsl:stylesheet>

Javascript

<table>
<thead>
<tr>
<th>Eco-parser embedded Javascript functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>var elems = document.getElementsByClassName(&quot;tiger&quot;);</td>
</tr>
<tr>
<td>var arr = [];</td>
</tr>
</tbody>
</table>
```javascript
var text = "";

// Location variables
var loc = "";
var lat = "";
var long = "";
var coords = "";
var locpage = "";
var gazref = "";

// Sighting variables
var sighting = "";
var show = "";
var clas = "";
var sipage = "";

// XMLHttpRequest
xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/newfile.pl", true);
// xhttp.send();

// Function to manually highlight text
function myFunction2() {
    if (window.getSelection) {
        text = window.getSelection().toString();
    } else if (document.selection && document.selection.type != "Control") {
        text = document.selection.createRange().text;
    }
    show = "[" + text + "]";
    myFunction6(show);
}

// Function for location confirmation buttons (shows coordinates)
function myFunction4(name,latitude,longitude,gaz,page,word, context) {
    loc = name
    lat = latitude
    long = longitude
    locpage = page
    gazref = gaz
    coords = "Location: " + name + "\nLatitude: " + lat + "\nLongitude: " + long + "\nGazateer reference: (" + gaz + ")" + "\nPage: " + page + "\nWord ID: " + word;
    if (confirm("Sighting selected: \nSighting selected: \nProposed sighting location: \nConfirm this sighting and location then send to output?\n\n") == true) {
        // triggers confirmation alert
        alert(coords + " \n\nCoordinates sent.");
        // triggers perl sending function
        loadDoc4();
    }
}
```

// Select sighting, grab context
function myFunction5(word, prec1, prec2, prec3, fol1, fol2, fol3, id, page) {
    context = """+prec1 + " " + prec2 + " " + prec3 + " " + word + " " + fol1 + " " + fol2 + " " + fol3+""
    sighting = """+word+"" from page: ""+page + "\n\nContext: " +context + "\n\n"
    alert(sighting);
}

function myFunction3(word, prec1, prec2, prec3, fol1, fol2, fol3, id, page) {
    classi = "Wild tiger sighting";
    sipage = page
    context = """+prec1 + " " + prec2 + " " + prec3 + " " + word + " " + fol1 + " " + fol2 + " " + fol3+""
    sighting = """+word+"" from page: ""+page + "\n\nContext: " +context + "\n\n"
    var clas = prompt("Is this classification correct?", classi);
    if (clas != null) {
        alert(clas + "\n" + sighting);
    }
}

function myFunction7(word, prec1, prec2, prec3, fol1, fol2, fol3, id, page) {
    classi = "Person related to tiger";
    sipage = page
    context = """+prec1 + " " + prec2 + " " + prec3 + " " + word + " " + fol1 + " " + fol2 + " " + fol3+""
    sighting = """+word+"" from page: ""+page + "\n\nContext: " +context + "\n\n"
    var clas = prompt("Is this classification correct?", classi);
    if (clas != null) {
        alert(clas + "\n" + sighting);
    }
}

function myFunction8(word, prec1, prec2, prec3, fol1, fol2, fol3, id, page) {
    classi = "Tiger related product";
    sipage = page
    context = """+prec1 + " " + prec2 + " " + prec3 + " " + word + " " + fol1 + " " + fol2 + " " + fol3+""
    sighting = """+word+"" from page: ""+page + "\n\nContext: " +context + "\n\n"
    var clas = prompt("Is this classification correct?", classi);
    if (clas != null) {
        alert(clas + "\n" + sighting);
    }
}

function myFunction9(word, prec1, prec2, prec3, fol1, fol2, fol3, id, page) {
    classi = "Tiger related simile";
    sipage = page
    context = """+prec1 + " " + prec2 + " " + prec3 + " " + word + " " + fol1 + " " + fol2 + " " + fol3+""
    sighting = """+word+"" from page: ""+page + "\n\nContext: " +context + "\n\n"
    var clas = prompt("Is this classification correct?", classi);
    if (clas != null) {
        alert(clas + "\n" + sighting);
    }
}
function myFunction10(word, prec1, prec2, prec3, fol1, fol2, fol3, id, page) {
    classi = "Captive tiger";
    sipage = page
    context = "" + prec1 + " " + prec2 + " " + prec3 + " " + word + " " + fol1 + " " + fol2 + " " + fol3 + "";
    sighting = "" + word + "" from page: " + page + "\nContext: " + context + "\n\n";
    var clas = prompt("Is this classification correct?", classi);
    if (clas != null) {
        alert(clas + "\n" + sighting);
    }
}

// Function to show, confirm and trigger sending of manually highlighted text for re-geoparsing
function myFunction6(show) {
    if (confirm("Sighting selected: \n" + sighting + "\nProposed sighting location to re-geoparse: \n" + show + "\n\nConfirm this sighting and location and send to output?\n") == true) {
        // triggers confirmation alert
        alert(show + "\n\nsent.");
        // triggers perl sending function
        loadDoc2();
    } else {
        x = "You pressed Cancel!";
    }
    document.getElementById("demo").innerHTML = x;
}

// Send encoded geojson feature collection to geojson.io for visualisation
function myFunction11() {
    // var url = "https://www.geos.ed.ac.uk/~s1581683/cgi-bin/send.pl";
    if (confirm("Visualise in geojson.io?") == true) {
        window.open("http://geojson.io/#data=data:text/x-url,"+url);
    }
}

// trigger script to create file for geojson feature collection to host points
function loadDoc1() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            document.getElementById("demo").innerHTML = xhttp.responseText;
        }
    };
    xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/newfile.pl", true);
    xhttp.send();
    alert("sent.");
}

// trigger script to end file for geojson feature collection that hosts points
function loadDoc5() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            document.getElementById("demo").innerHTML = xhttp.responseText;
        }
    }
    xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/endfile.pl", true);
    xhttp.send();
    alert("sent.");
    myFunction11();
}

// Function to send manually highlighted text to Perl
function loadDoc2() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            document.getElementById("demo").innerHTML = xhttp.responseText;
        }
    }
    xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/link.pl?fname=\"+encodeURIComponent(text), true);
    xhttp.send();
}

// Function to send coordinates
function loadDoc3() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            document.getElementById("demo").innerHTML = xhttp.responseText;
        }
    }
    xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/link.pl?fname=\"+encodeURIComponent(coords), true);
    xhttp.send();
}

// Function to send coordinates to perl script for geojson encoding
function loadDoc4() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (xhttp.readyState == 4 && xhttp.status == 200) {
            document.getElementById("demo").innerHTML = xhttp.responseText;
        }
    }
    xhttp.open("GET", "http://www.geos.ed.ac.uk/~s1581683/cgi-bin/geojson.pl?fname=\"+encodeURIComponent(loc)+"&lat=\"+encodeURIComponent(lat)+"&long=\"+encodeURIComponent(long)+"&locpage=\"+encodeURIComponent(locpage)+"&gazref=\"+encodeURIComponent(gazref), true);
    xhttp.send();
}
Perl

### Create new geoJSON file

```perl
#!/usr/bin/env perl

use CGI;
use strict;
use warnings;

my $cgi = CGI->new;
my $file = "/web/s1581683/public_html/dynamic/parser_points.geojson";

open(FILE, ">$file") or die ("Unable to open file $file : $!");
print FILE "{" 
    "type": "FeatureCollection", 
    "features": [
"
"; close(FILE);
print "Content-type: text/plain\n\n";
```

### Populate geoJSON file

```perl
#!/usr/bin/env perl

use CGI;
use strict;
use warnings;

my $cgi = CGI->new;

my $name = $cgi->param('fname');
my $lat = $cgi->param('lat');
my $long = $cgi->param('long');
my $locpage = $cgi->param('locpage');
my $gazref = $cgi->param('gazref');
my $clas = $cgi->param('clas');
my $sipage = $cgi->param('sipage');
#my $lat = 125.6;
#my $long = 10.1;
#my $name = "bob";

my $file = "/web/s1581683/public_html/dynamic/parser_points.geojson";

#open(FILE, "<$file") or die ("Unable to open file $file : $!");
#print FILE "\n";
```
#!/usr/bin/env perl
use CGI;
#use strict;
use warnings;

my $cgi = CGI->new;
my $file = "/web/s1581683/public_html/dynamic/parser_points.geojson";
my $size = -s $file;
open(FILE, "+<", $file) or die ("Unable to open file $file : $!");
seek FILE, $size-2, 0;
print FILE "\n"
print FILE "}

print FILE "End geoJSON file"

Format geoJSON file for geojson.io
#!/usr/bin/env perl
use CGI;

my $cgi = CGI->new;
my $file = "/web/s1581683/public_html/dynamic/parser_points.geojson";
my $size = -s $file;
open(FILE, "+<", $file) or die ("Unable to open file $file : $!");
seek FILE, $size-2, 0;
print FILE "\n"
print FILE "}

print "Content-type: text/plain\n\n";
#!/usr/bin/env perl

use CGI;
use strict;
use warnings;

my $cgi = CGI->new;
my $file = "/web/s1581683/public_html/dynamic/parser_points.geojson";

local $/;

print "Access-Control-Allow-Origin: *
)n"

open(FILE, $file) or die "Can't read file 'filename' [$!]
)n";
my $document = <FILE>;
close(FILE);
print $document;

Send extracted data to text file

#!/usr/bin/env perl

use CGI;
use strict;
use warnings;

my $cgi = CGI->new;
my $name = $cgi->param('fname');

my $file = "/web/s1581683/public_html/dynamic/test.txt";

open(FILE, ">>$file") or die ("Unable to open file $file : $!");
print FILE $name."\n"
close(FILE);

print "Content-type: text/plain\n\n"
print $name;

Thesaurus Preparation (XSLT)

<XSLT version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
exclude-result-prefixes="owl skos">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
  <Groups>
    <xsl:apply-templates select="*"/>
  </Groups>
</xsl:template>

<xsl:template match="*">
  <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template match="owl:NamedIndividual">
  <xsl:apply-templates select="skos:altLabel|skos:prefLabel"/>
</xsl:template>

<xsl:template match="skos:altLabel|skos:prefLabel">
  <Group>
    <xsl:text> </xsl:text>
    <xsl:text>&#09;</xsl:text>
    <Name1><xsl:value-of select="node()" /></Name1>
    <xsl:text>&#10;</xsl:text>
    <xsl:text>&#09;</xsl:text>
    <Name2><xsl:value-of select="substring-after(../@rdf:about, '#')" /></Name2>
    <xsl:text>&#10;</xsl:text>
    <xsl:text>&#09;</xsl:text>
    <Name3><xsl:value-of select="substring-after(../skos:broader/@rdf:resource, '#')" /></Name3>
  </Group>
</xsl:template>

</xsl:stylesheet>

---

String lowercase

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="smallCase" select="'abcdefghijklmnopqrstuvwxyz'"/>
  <xsl:variable name="upperCase" select="'ABCDEFGHIJKLMNOPQRSTUVWXYZ'"/>

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="text()">
    <xsl:value-of select="translate(., $upperCase, $smallCase)"/>
  </xsl:template>

</xsl:stylesheet>
Thesaurus (SKOS format)

Tiger term thesaurus

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns="file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#"
    xml:base="file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:undefined="undefined#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/rdf-syntax-ns#"
    xmlns:skos="http://www.w3.org/2004/02/skos/core#">
    <owl:Ontology rdf:about="file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined"/>

    <!-- undefined#audience -->
    <owl:AnnotationProperty rdf:about="undefined#audience"/>

    <!-- undefined#created -->
    <owl:AnnotationProperty rdf:about="undefined#created"/>

    <!-- undefined#creator -->
    <owl:AnnotationProperty rdf:about="undefined#creator"/>

    <!-- undefined#description -->
    <owl:AnnotationProperty rdf:about="undefined#description"/>
```

<!-- undefined#language -->
<owl:AnnotationProperty rdf:about="undefined#language"/>

<!-- undefined#title -->
<owl:AnnotationProperty rdf:about="undefined#title"/>

<!-- undefined#version -->
<owl:AnnotationProperty rdf:about="undefined#version"/>

<!--
 // Object Properties
 // //////////////////////////////////////////////////////////////////////////////////////////
 -->

<!-- http://www.w3.org/2004/02/skos/core#broader -->
<owl:ObjectProperty rdf:about="http://www.w3.org/2004/02/skos/core#broader"/>

<!-- http://www.w3.org/2004/02/skos/core#hasTopConcept -->
<owl:ObjectProperty rdf:about="http://www.w3.org/2004/02/skos/core#hasTopConcept"/>

<!-- http://www.w3.org/2004/02/skos/core#inScheme -->
<owl:ObjectProperty rdf:about="http://www.w3.org/2004/02/skos/core#inScheme"/>

<!-- http://www.w3.org/2004/02/skos/core#narrower -->
<owl:ObjectProperty rdf:about="http://www.w3.org/2004/02/skos/core#narrower"/>

<!--
///////////////////////////////////////////////////////////////////////
// Data properties
///////////////////////////////////////////////////////////////////////
-->

<!-- file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#audience -->
<owl:DatatypeProperty rdf:about="file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#audience"/>

<!-- file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#created -->
<owl:DatatypeProperty rdf:about="file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#created"/>

<!-- file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#creator -->
<owl:DatatypeProperty rdf:about="file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#creator"/>

<!-- file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#description -->
<owl:DatatypeProperty rdf:about="file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#description"/>

<!-- file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#language -->
<owl:DatatypeProperty rdf:about="file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#language"/>

<!-- file:/opt/liferay/liferay-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#title -->
// Individuals
//

<!-- file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#TigerThesaurus -->

<owl:NamedIndividual rdf:about="file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#TigerThesaurus">
  <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#ConceptScheme"/>
  <undefined:language>en</undefined:language>
  <undefined:audience></undefined:audience>
  <undefined:version>1</undefined:version>
  <undefined:created>14/05/2016</undefined:created>
  <undefined:description>Thesaurus of tiger terms</undefined:description>
  <undefined:creator>Ciaran Hoy</undefined:creator>
  <undefined:title>TigerThesaurus</undefined:title>
  <skos:hasTopConcept rdf:resource="http://null/resource#Beast"/>
  <skos:hasTopConcept rdf:resource="http://null/resource#Cat"/>
  <skos:hasTopConcept rdf:resource="http://null/resource#Tiger"/>
</owl:NamedIndividual>

<!-- http://null/resource#Beast -->

<owl:NamedIndividual rdf:about="http://null/resource#Beast">
  <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
  <skos:inScheme rdf:resource="file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#TigerThesaurus"/>
  <skos:narrower rdf:resource="http://null/resource#Cat"/>
</owl:NamedIndividual>

<!-- http://null/resource#Cat -->

<owl:NamedIndividual rdf:about="http://null/resource#Cat">
  <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
  <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Big cat</skos:altLabel>
  <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Cub</skos:altLabel>
  <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Feline</skos:altLabel>
</owl:NamedIndividual>
<skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Great cat</skos:altLabel>
 <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Large cat</skos:altLabel>
 <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Striped cat</skos:altLabel>
 <skos:inScheme rdf:resource="file:/opt/liferay/liferay-portal-6.2.0-ce-rc6/tomcat-7.0.42/temp/undefined#TigerThesaurus"/>
 <skos:broader rdf:resource="http://null/resource#Beast"/>
 <skos:narrower rdf:resource="http://null/resource#Tiger"/>
</owl:NamedIndividual>

<!-- http://null/resource#Tiger -->

<owl:NamedIndividual rdf:about="http://null/resource#Tiger">
 <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
 <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Felis tigris</skos:altLabel>
 <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">P. t.</skos:altLabel>
 <skos:altLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">P.t.</skos:altLabel>
 <skos:prefLabel rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Panthera tigris</skos:prefLabel>
 <skos:broader rdf:resource="http://null/resource#Cat"/>
</owl:NamedIndividual>

<!-- Generated by the OWL API (version 3.4.4) http://owlapi.sourceforge.net -->
<Groups xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <Group>
    <Name1>tiger</Name1>
    <Name2>tigerthesaurus</Name2>
    <Name3/>
  </Group>
  <Group>
    <Name1>big cat</Name1>
    <Name2>cat</Name2>
    <Name3>beast</Name3>
  </Group>
  <Group>
    <Name1>cub</Name1>
    <Name2>cat</Name2>
    <Name3>beast</Name3>
  </Group>
  <Group>
    <Name1>feline</Name1>
    <Name2>cat</Name2>
    <Name3>beast</Name3>
  </Group>
  <Group>
    <Name1>great cat</Name1>
    <Name2>cat</Name2>
    <Name3>beast</Name3>
  </Group>
  <Group>
    <Name1>large cat</Name1>
    <Name2>cat</Name2>
    <Name3>beast</Name3>
  </Group>
  <Group>
    <Name1>striped-cat</Name1>
    <Name2>cat</Name2>
    <Name3>beast</Name3>
  </Group>
  <Group>
    <Name1>felis tigris</Name1>
    <Name2>tiger</Name2>
    <Name3>cat</Name3>
  </Group>
  <Group>
    <Name1>p. t.</Name1>
    <Name2>tiger</Name2>
    <Name3>cat</Name3>
  </Group>
  <Group>
    <Name1>p.t.</Name1>
    <Name2>tiger</Name2>
    <Name3>cat</Name3>
  </Group>
</Groups>
Exclusion lists

<table>
<thead>
<tr>
<th>Product exclusion list</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;</td>
</tr>
<tr>
<td>&lt;List xmlns:rdf=&quot;http://www.w3.org/1999/02/22-rdf-syntax-ns#&quot;&gt;</td>
</tr>
<tr>
<td>&lt;Phrase id=&quot;x1&quot;&gt;bone&lt;/Phrase&gt;</td>
</tr>
<tr>
<td>&lt;Phrase id=&quot;x2&quot;&gt;bones&lt;/Phrase&gt;</td>
</tr>
<tr>
<td>&lt;Phrase id=&quot;x3&quot;&gt;skin&lt;/Phrase&gt;</td>
</tr>
<tr>
<td>&lt;Phrase id=&quot;x4&quot;&gt;skins&lt;/Phrase&gt;</td>
</tr>
<tr>
<td>&lt;Phrase id=&quot;x5&quot;&gt;skull&lt;/Phrase&gt;</td>
</tr>
<tr>
<td>&lt;Phrase id=&quot;x6&quot;&gt;skulls&lt;/Phrase&gt;</td>
</tr>
<tr>
<td>&lt;/List&gt;</td>
</tr>
</tbody>
</table>
<?xml version="1.0" encoding="UTF-8"?>
<List xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<Phrase id="x1">tame</Phrase>
<Phrase id="x2">tamed</Phrase>
<Phrase id="x3">captive</Phrase>
<Phrase id="x4">caged</Phrase>
<Phrase id="x5">enclosure</Phrase>
<Phrase id="x6">trapped</Phrase>
<Phrase id="x6">fenced</Phrase>
</List>

<?xml version="1.0" encoding="UTF-8"?>
<List xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<Phrase id="x1">hunter</Phrase>
<Phrase id="x2">observer</Phrase>
<Phrase id="x3">tracker</Phrase>
<Phrase id="x4">hunting</Phrase>
<Phrase id="x5">tracking</Phrase>
<Phrase id="x6">stalker</Phrase>
<Phrase id="x7">stalking</Phrase>
<Phrase id="x8">poacker</Phrase>
</List>