This contextual representation of a word is used in **automated word sense disambiguation**: current context of an ambiguous word is compared with “typical” contexts of each possible sense

Also, word representation by a **word context vector** is related to word embedding (Google’s word2vec): vector is “implicit” vector learned to optimize prediction of co-occurrences of words in a large corpus instead of an explicit context (for word embeddings, see Section 14.3.3 of the book)

(from Exam2 Review Sheet).

See summarized lectures on text categorization (slides w/models)

**Converting html to text**:

### Using Lynx

Another text-based browser that could be used for this task is [Lynx](http://lynx.isc.org/). Like w3m, it has a non-interactive mode that is selected by the option -dump:

lynx -dump input.html > output.txt

The output width defaults to 80 characters, and the output encoding defaults to ISO-8859-1. The width can be changed using the -width option:

lynx -dump -width 120 input.html > output.txt

and the encoding using the -display\_charset option:

lynx -dump -display\_charset UTF-8 input.html > output.txt

### Alternatives: w3m and [Links](http://www.jikos.cz/~mikulas/links/). A

**Voiced VideoggGGgggg**

Camtasia

Activepresenter is also good: <https://atomisystems.com/download/>

**Package installed, but PyCharms & anaconda do not see it**:

reinstall with “conda install -c anaconda package\_name”

Ismini on project:

So basically, no matter the sense (meaning) of a term, i.e. whether "bank" in that specific case means "financial institution" or "river" you will get a huge list of mentions, and most likely the majority of those refer to "financial institution" instead of "river"

You could:

1. cluster by sense.

**Wordnet** is a tool that contains r**elations between words, such as synonyms, hypernyms,** etc.

It also provides the **difference senses of a word** and short examples of usage.

You can use a **TFIDF representation** or better yet, use **pretrained word embeddings**, for both the example of usage and the phrases that the current tool returns. Then calculate the **cosine similarity** for each phrase and the **sense example from Wordnet**, thus you will assign each phrase to a particular sense. Finally, with another metric, choose the most representative example of each sense to show to the user.

[the tool is available in many languages](http://globalwordnet.org/wordnets-in-the-world/)

2. output some kind of time-series analysis on the different senses. If you have data for a set of years, then how is the term usage changing across time? I imagine that "bank" was mostly used as a "river" at some point and suddenly became prevalent as "financial institution"

In general, modelling such dynamics is very compelling for understanding how language evolves. Perhaps by understanding, we might find ways to drive language where we would like, for example alleviate the popularity of abbreviations and slang (can't really read twitter posts these days!) or even find out how we learn language and pass it on to computers.

Sites that may be useful for the Project:

0a) THEORY  
Text classification algorithms in data mining: <http://www.expertsystem.com/text-classification-algorithms-data-mining/>

0b) GREAT GENERAL DATA SCIENCE resource with **examples**!

<https://towardsdatascience.com/>

**Example of data scientist portfolio**: <https://github.com/sajal2692/data-science-portfolio>

**Data Scientist Resume Projects**: <https://blog.statsbot.co/data-scientist-resume-projects-806a74388ae6>  
(ML problems set to build a data scientist CV **without work experience**)

**Pandas** <https://towardsdatascience.com/how-to-learn-pandas-108905ab4955>

**ML and AI!** <https://machinelearnings.co/>

**ML and analytics**: <https://blog.statsbot.co/> (lots of Russian authors)

Some ideas <https://blog.statsbot.co/text-classifier-algorithms-in-machine-learning-acc115293278>

CLASSICAL PYTHON

1) <https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>

**Machine Learning, NLP: Text Classification using scikit-learn, python and NLTK!!!!**

2) [http://textblob.readthedocs.io/en/dev/classifiers.html#](http://textblob.readthedocs.io/en/dev/classifiers.html)  
Tutorial: Building a Text Classification System with TextBlob

2a) <http://stevenloria.com/how-to-build-a-text-classification-system-with-python-and-textblob/>  
Tutorial: Simple Text Classification with Python and TextBlob

As an offshoot of 1a) – this is an interesting feature: Add color to my terminal <http://killtheyak.com/add-color-to-terminal/>. See also other no-BS guides on killtheyak.com

TextBlob\* stands on the giant shoulders of NLTK and pattern\*\*, and plays nicely with both.

**Features**

* Noun phrase extraction
* Part-of-speech tagging
* Sentiment analysis
* Classification (Naive Bayes, Decision Tree)
* Language translation and detection powered by Google Translate
* Tokenization (splitting text into words and sentences)
* Word and phrase frequencies
* Parsing
* n-grams
* Word inflection (pluralization and singularization) and lemmatization
* Spelling correction
* Add new models or languages through extensions
* WordNet integration

\* <http://textblob.readthedocs.io/en/dev/index.html>  
TextBlob *is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.*

\*\* <https://www.clips.uantwerpen.be/pages/pattern-en>  
*The pattern.en Python module contains a fast part-of-speech tagger for English (identifies nouns, adjectives, verbs, etc. in a sentence), sentiment analysis, tools for English verb conjugation and noun singularization & pluralization, and a WordNet interface*

3) **Cogito Intelligence API** (includes semantic analysis!)

Key Features

* Content Categorization with specialized classification for intelligence taxonomy, crime taxonomy, terrorism taxonomy, cyber crime taxonomy, geographic taxonomy
* Entity, Time References (alpha version), Facts and Relationships Extraction, over 50 domain entities (world leaders, terrorist organizations, organized crime, unconventional weapons, geographic locations, etc.)
* Writeprint for stylometric characterization and authorship assessment
* Emotions detection (anger, fear, disgust, sadness, happiness, surprise, etc.) related to the entities within the text
* Social Metadata Disambiguation (e.g. #bringbackourgirls transforms into [bring back our girls])
* Semantic Reasoning for surfacing “hidden” entities
* Writeprint for stylometric characterization and authorship assessment

This **free web service includes all the 16 channels** available: full categorization, intelligence categorization, crime & offense categorization, terrorism categorization, cyber-crime categorization, geographic categorization, emotions, people text mining, organization text mining, place text mining, domain entities text mining, fact mining, summary, writeprint and semantic relations between entities.

Web service calls are limited to **one every 100 seconds**.

The free option is limited to **two months of service**.

4) <https://en.wikipedia.org/wiki/Weka_(machine_learning)>  
Weka - a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions

5) <https://www.meaningcloud.com/developer/text-classification>

A trainable model. Text Classification is MeaningCloud's solution for automated document classification. It assigns one or more categories to a text, using standard domain-specific taxonomies (e.g., IPTC. IAB, ICD-10) or user-defined categories. The algorithm combines statistical document classification with rule-based filtering, which allows to obtain a high degree of accuracy and flexibility in a wide range of environments.

Differentiators:

* Statistical classifiers provide a means to use example documents to define each category.
* In turn, rule base classifiers helps to fine-tune the classification and correct the output of statistical classifiers.
* Its powerful rule based classification language is also useful to bootstrap a categorization when no examples are available.
* Features predefined, standard taxonomies: IPTC, IAB, ICD-10, Eurovoc.
* User can create his/her own categories and classification models.

6) <http://www.opencalais.com/about-open-calais/> (does extract some relevant topics, but several)  
Open Calais (has a free API) is a free service currently accessible via a public website (opencalais.com) and will also be available via a Thomson Reuters sponsored public website called PermID.org. This free service provides document tagging using an extensive set of fields such as Company, Person, Geography, Industry Classifications, Topics, Social Tags, Facts, and Events. The service is hosted by Thomson Reuters and allows users to upload up to 5,000 documents a day.

Open Calais is a sophisticated Thomson Reuters web service that attaches intelligent metadata-tags to your unstructured content, enabling powerful text analytics. The Open Calais natural language processing engine automatically analyzes and tags your input files in such a way that your consuming application can both easily pinpoint relevant data, and effectively leverage the invaluable intelligence and insights contained within the text.

Open Calais analyzes the semantic content of your input files using a combination of statistical, machine-learning, and custom pattern-based methods. Developed by the Text Metadata Services (TMS) group at Thomson Reuters, Open Calais outputs highly accurate and detailed metadata. Open Calais also maps your metadata-tags to Thomson Reuters unique IDs. This supports disambiguation (and linking) of data across all documents processed by Open Calais, and also offers you the opportunity to further enrich your data with related information from the Thomson Reuters datasets.

Open Calais automatically analyzes your input text and performs the following processes:

• Named Entity and Relationship Recognition – Open Calais identifies and tags mentions (text strings) of things like companies, people, deals, geographical locations, industries, physical assets, organizations, products, events, etc., based on a list of predefined metadata types.

• Aboutness Tagging – Open Calais assigns social, topic, and industry tags that describe what the input document is about as a whole.

7) Check this Java module as well:MALLET (MAchine Learning for LanguagE Toolkit)

<http://mallet.cs.umass.edu/>

MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text.

MALLET includes sophisticated tools for **document classification**: efficient routines for converting text to "features", a wide variety of algorithms (including Naïve Bayes, Maximum Entropy, and Decision Trees), and code for evaluating classifier performance using several commonly used metrics. [[Quick Start](http://mallet.cs.umass.edu/classification.php)] [[Developer's Guide](http://mallet.cs.umass.edu/classifier-devel.php)]

In addition to classification, MALLET includes tools for **sequence tagging** for applications such as named-entity extraction from text. Algorithms include Hidden Markov Models, Maximum Entropy Markov Models, and Conditional Random Fields. These methods are implemented in an extensible system for finite state transducers. [[Quick Start](http://mallet.cs.umass.edu/sequences.php)] [[Developer's Guide](http://mallet.cs.umass.edu/fst.php)]

Topic models are useful for analyzing large collections of unlabeled text. The MALLET **topic modeling** toolkit contains efficient, sampling-based implementations of Latent Dirichlet Allocation, Pachinko Allocation, and Hierarchical LDA. [[Quick Start](http://mallet.cs.umass.edu/topics.php)]

Many of the algorithms in MALLET depend on **numerical optimization**. MALLET includes an efficient implementation of Limited Memory BFGS, among many other optimization methods. [[Developer's Guide](http://mallet.cs.umass.edu/optimization.php)]

In addition to sophisticated Machine Learning applications, MALLET includes routines for transforming text documents into numerical representations that can then be processed efficiently. This process is implemented through a flexible system of "pipes", which handle distinct tasks such as tokenizing strings, removing stopwords, and converting sequences into count vectors. [[Quick Start](http://mallet.cs.umass.edu/import.php)] [[Developer's Guide](http://mallet.cs.umass.edu/import-devel.php)]

An add-on package to MALLET, called GRMM, contains support for inference in general graphical models, and training of CRFs with arbitrary graphical structure. [[About GRMM](http://mallet.cs.umass.edu/grmm/index.php)]

8) Zotero citation management tool

<https://www.zotero.org/>