Short overview on newest trends and development directions of Natural Language Processing in Haskell

Alina Parascan

Alexandru Velicea

1. What is NLP (Natural Language Processing) ?

As defined by Wikipedia " **Natural language processing** (**NLP**) is a field of computer science, artificial intelligence and linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input."

A short insight of NLP starts with the very beginning of computers. In 1950 Alan Turing published an article called "Computing Machinery and Intelligence" which proposed what is now known as the "Turing Test" as a criterion of intelligence. In reality, he made a breakthrough in this field much earlier, during the second World War when he conceived and built a machine that helped decrypt the “Enigma” code used by the Germans to pass secret military messages. This was one of the first computers and of course the first instance in which a computer was used for Natural Language Processing purposes. Its importance is proved by the fact that decrypting the “Enigma” code shortened the war with 2 to 4 years and was estimated to saving the lives of over 4 million people [1]

2. Uses of NLP libraries

There are numerous uses of NLP libraries: keyword generation, language identification, full-text search, artificial intelligence, auto completion, translation, topic detection and clustering, content summarization, parsing human-written dates and locations, etc.

We will mention some of the most notable Major Tasks in NLP:

Automatic summarization, which produces a short and readable summary of a single or multiple documents, preserving the relevant information.[2]

Machine translation, which means to automatically translate a text from one human language to another. Putting aside Google Translate, some notable software in this field are Babylon, Power Translator, Prompt[3]

[1] Jack Copeland, (18 June 2012). "Alan Turing: The codebreaker who saved 'millions of lives'". BBC News Technology. Retrieved 26 October 2014

[2] Dipanjan Das, André F.T. Martins, “A Survey on Automatic Text Summarization”, Carnegie Mellon University, November 2007

[3] according to http://translation-software-review.toptenreviews.com/

Named entity recognition (NER), which is the task of correctly identification of proper names of people, organizations, locations, or other entities, determining in the same time the type of each such name is (e.g. person, location, organization). [4]

Natural language generation, which convert information from computer databases into readable human language, thus deliberately constructing a natural language text in order to meet specified communicative goals.[5]

Part-of-speech tagging, which is the process of classifying words into their **parts of speech** and labeling them accordingly.[6] As we know, many words, especially common ones, can serve as multiple parts of speech. For example, "color" can be a noun ("the color of the sky is blue") or verb ("to color a drawing"); "stone" can be a noun, verb (to stone) or adjective (a stone bridge); and "out" can be any of at least five different parts of speech. Some languages have more such ambiguity than others. English language has little inflectional morphology, therefore is particularly prone to such ambiguity. Chinese is prone to such ambiguity because it is a tonal language during verbalization. Such inflection is not readily conveyed via the entities employed within the orthography to convey intended meaning.

Parsing - determines the parse tree (grammatical analysis) of a given sentence.

Question answering – determines the answer of a human-language question. Point wise questions have a specific correct answer (such as "What is the capital of Romania?"), but

[4] Benjamin Bengfort, "An Introduction to Named Entity Recognition in Natural Language Processing - Part 1", 17 April 2013, www.datacommunnitydc.org

[5] Claire Gardent, "Natural Language Generation", CNRS/LORIA, Nancy, http://www.loria.fr/~gardent/teaching/nlg-bkk06.pdf

[6] Steven Bird, Ewan Klein, and Edward Loper, "Natural Language Processing", O'Reilly Media, 2009

sometimes the question considered is an open-ended one (such as "What is the definition of happiness?").

Sentence breaking (also known as sentence boundary disambiguation) - finds the sentence boundaries in a given piece of text. Sentence boundaries are often marked by period, commas or other punctuation marks, but these same characters can serve other purposes (e.g. marking abbreviations).

Sentiment analysis - extract subjective information usually from a set of documents, often using online reviews to determine the contextual polarity of specific objects in a text. In other words, it determines whether a piece of writing is positive, negative or neutral.[7] It is especially useful for identifying trends of public opinion in the social media, for the purpose of marketing.

Among other notable and recent uses of NLP libraries we would like to mention the work of Hakan Burden, Rogardt Heldal and Peter Ljungl from University of Gothenburg, Sweden, "*Opportunities for Agile Documentation Using Natural Language Generation*", that discuss the opportunities for Agile using Natural Language Generation in order to prevent the general miscommunication that occurs between the client and the actual developers and to realize a more accurate approach to requirements specifications. They propose Natural Language Generation as a suitable framework for automatically retrieving relevant information from the source code of certain modules within the application that will impact future tasks. This information will be then displayed in an appropriate text format (i.e. adding the necessary HTML tags or in a LATEX format for the presentation). Their conclusion was:

> "Using the code for generating documents that describe different system properties enables a single source of information in the development process, aligning documentation with implementation. Different views could then be generated from the code during implementation, with content and structure

[7] www.semantria.com "What is Sentiment Analysis?"

depending on who the consumer of the document and what the purpose of the text is. This could be done together with the relevant stakeholders, enabling concise documents that are easy to keep consistent with the implementation. The opportunities for collaborations between industry and academia in text generation as a means for Agile documentation are abundant! "

These are a few of the most important and recent uses of NLP, most of which have already been implemented and used, but considering the continuous growth of the volume of information present on the internet and the increasing importance of social media in day to day life, we are convinced that NLP will be used in effective ways to sort, search and interpret vast volumes of data/information.

3. NLP and Haskell

Currently there are over 70 NLP libraries available on Hackage and this speaks volumes in regards to the direction and evolution of functional programming in general, Haskell in particular and of course the increasing attention that Natural Language Processing is getting recently. Even though the basic algorithms and principles used for NLP were developed a while ago, some of them have evolved and have been improved through the years and are still used. One of those is the Porter Stemmer Algorithm that was originally written in 1979 at Cambridge University. The purpose of this algorithm is to reduce words to common roots; for example it should identify the string "cats" or "catlike" as based on the root "cat" and "stemmer", "stemming", "stemmed" as based on "stem". This is particularly useful when building a search engine, or a language translation tool, or for any Natural Language Processing purpose as reducing words to their root makes processing them easier.

The Haskell version of this algorithm was written by Dimitry and it is still used present day.

Another important tool in the Natural Language Processing world is LOLITA. The name is an acronym for "**L**arge-scale, **O**bject-based, **L**inguistic **I**nteractor, **T**ranslator and **A**nalyzer". It is an NLP system that provides core capabilities such as:

1. " conversion of English language text to an internal (semantic network based) representation of the text's meaning
2. inferences in the semantic network
3. conversion of portions of the semantic network to English "[8]

"LOLITA was an early example of a substantial application written in a functional language: it consisted of around 50,000 lines of Haskell, with around 6000 lines of C. It is also a complex and demanding application, in which many aspects of Haskell were invaluable in development.
LOLITA was designed to handle unrestricted text, so that ambiguity at various levels was unavoidable and significant. Laziness was essential in handling the explosion of syntactic ambiguity resulting from a large grammar, and it was much used with semantic ambiguity too." [9]

[8] http://homepages.inf.ed.ac.uk/wadler/realworld/natlangproc.html

[9] http://en.wikipedia.org/wiki/LOLITA

4. WordNet

Of course, in order to utilize NLP capabilities to the fullest one needs different helping tools; for example a database that contains words of a specific language in order to be able to process text and information. One of the most significant databases of this sort is WordNet. It is a lexical database for the English language and it was originally developed in 1985 in the Cognitive Science Laboratory at Princeton University under the direction of psychology professor George Armitage Miller.

WordNet groups English words into sets of synonyms (also called synsets) and it provides some short definitions and also records relations among these synonyms sets and their members. The database includes lexical categories such as: nouns, verbs, adjectives and adverbs but it disregards prepositions and other function words. It is primarily used in artificial intelligence and text analysis and the most recent version contains 155,287 words that are organized in 117,659 synsets and thus holding a total of 206,941 word-sentence pairs.

WordNet has been used for a number of different purposes such as automatic text classification, text summarization, machine translation, and even automatic crosswords puzzle generation.

Even though it was first developed in 1985 it is still currently used and interfaces in different programming languages have been developed for WordNet throughout the years. For example, one of the first Haskell interfaces for WordNet has been written in 2003 by Hal Daumé and after that two more have been written; one in 2008: it is available on Hackage at *http://hackage.haskell.org/package/WordNet-0.1.2* and the most recent Haskell Interface for this database was written in 2013 (also available on Hackage - *https://hackage.haskell.org/package/WordNet*).

This proves the increasing interest in recent years for text processing and Natural Language Processing and also that functional programming is one of the main programming styles that are and will be used for these types of endeavors as more and more libraries and tools are developed in order to support the growing need and interest for such subjects.

5. Practical examples of NLP libraries and applications written in Haskell

There are numerous libraries for Natural Language Processing available on Hackage and quite a few recent applications. We have chosen to start with a short presentation of the Chatter NLP Library. [10]

It is of course open source and available for download and usage and it comprises NLP algorithms that are developed for the purpose of tagging parts of speech according to the Brown Corpus. [11]

After installing it with the aid of `cabal install` command the user can input a text and the tagger function will output a list of every word that was previously input and tag each word into a category according with Brown Corpus list. For example the text "The best jokes have no punchline." will be analyzed as follows:

```
[TaggedSent [
POS {posTag = DT, posToken = Token "The"},       //DT - means singular
                                                  determiner/quantifier (this, that)
POS {posTag = JJS, posToken = Token "best"},     //JJS - semantically superlative
                                                  adjective (chief, top)
POS {posTag = NNS, posToken = Token "jokes"},    //NNS - plural noun
```

[10] http://hackage.haskell.org/package/chatter

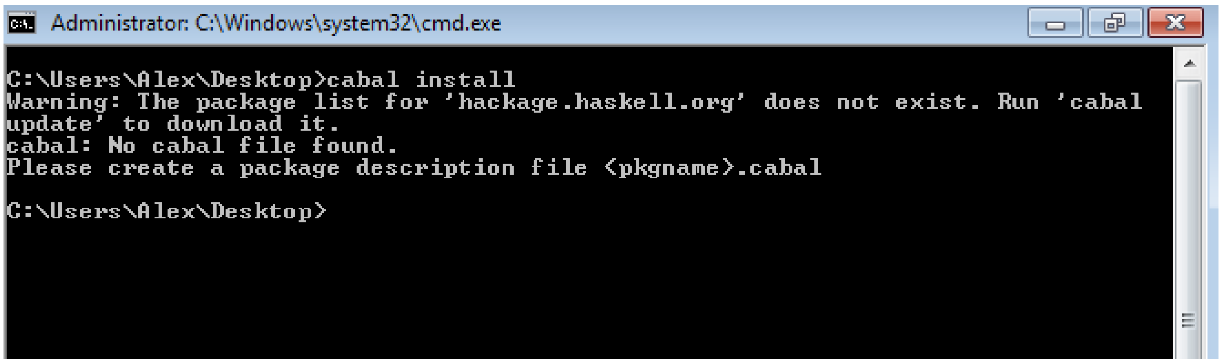
[11] http://en.wikipedia.org/wiki/Brown_Corpus#Part-of-speech_tags_used.

POS {posTag = VBP, posToken = Token "have"},
POS {posTag = DT, posToken = Token "no"},
POS {posTag = NN, posToken = Token "punchline"},
POS {posTag = Term, posToken = Token "."}]]

We will describe a tutorial on how to install and utilize chatter:

Step 1 : Install cabal

Open a cmd



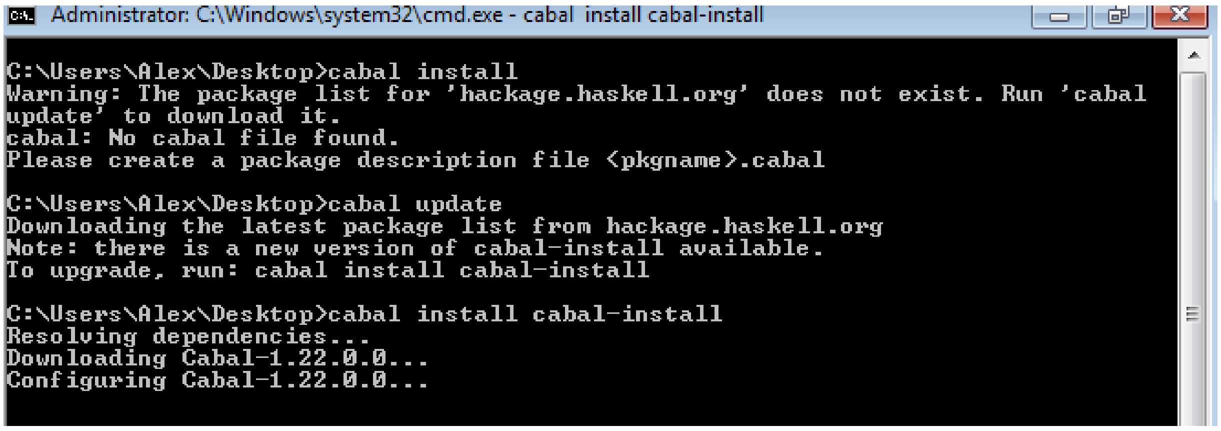
```
C:\Users\Alex\Desktop>cabal install
Warning: The package list for 'hackage.haskell.org' does not exist. Run 'cabal
update' to download it.
cabal: No cabal file found.
Please create a package description file <pkgname>.cabal

C:\Users\Alex\Desktop>
```

and type:

>cabal install

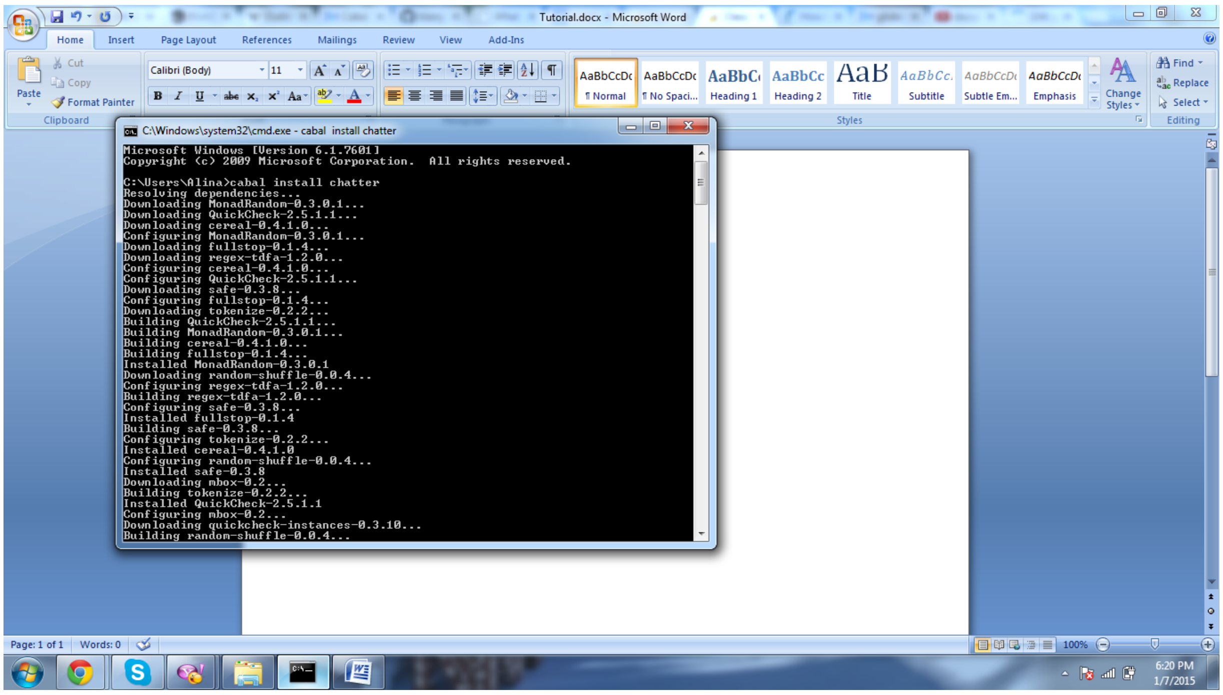
>cabal update

>cabal install cabal-install



```
C:\Users\Alex\Desktop>cabal install
Warning: The package list for 'hackage.haskell.org' does not exist. Run 'cabal
update' to download it.
cabal: No cabal file found.
Please create a package description file <pkgname>.cabal

C:\Users\Alex\Desktop>cabal update
Downloading the latest package list from hackage.haskell.org
Note: there is a new version of cabal-install available.
To upgrade, run: cabal install cabal-install

C:\Users\Alex\Desktop>cabal install cabal-install
Resolving dependencies...
Downloading Cabal-1.22.0.0...
Configuring Cabal-1.22.0.0...
```

Step two: Install chatter from Hackage:



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Alina>cabal install chatter
Resolving dependencies...
Downloading MonadRandom-0.3.0.1...
Downloading QuickCheck-2.5.1.1...
Downloading cereal-0.4.1.0...
Configuring MonadRandom-0.3.0.1...
Downloading fullstop-0.1.4...
Downloading regex-tdfa-1.2.0...
Configuring cereal-0.4.1.0...
Configuring QuickCheck-2.5.1.1...
Downloading safe-0.3.8...
Configuring fullstop-0.1.4...
Downloading tokenize-0.2.2...
Building QuickCheck-2.5.1.1...
Building MonadRandom-0.3.0.1...
Building cereal-0.4.1.0...
Building fullstop-0.1.4...
Installed MonadRandom-0.3.0.1
Downloading random-shuffle-0.0.4...
Configuring regex-tdfa-1.2.0...
Building regex-tdfa-1.2.0...
Configuring safe-0.3.8...
Installed fullstop-0.1.4
Building safe-0.3.8...
Configuring tokenize-0.2.2...
Installed cereal-0.4.1.0
Configuring random-shuffle-0.0.4...
Installed safe-0.3.8
Downloading mbox-0.2...
Building tokenize-0.2.2...
Installed QuickCheck-2.5.1.1
Configuring mbox-0.2...
Downloading quickcheck-instances-0.3.10...
Building random-shuffle-0.0.4...
```

Step three : Create a Haskell file named Main.hs and paste the following code :

1. Import the parts of speech library and the pack function:

```
import NLP.POS

import Data.Text (pack)
```

2. Obtain the default tagger provided by the library:

```
main = do

tagger <- defaultTagger
```

3. Feed the tag function a tagger and a text to see the corresponding parts of speech per each word:

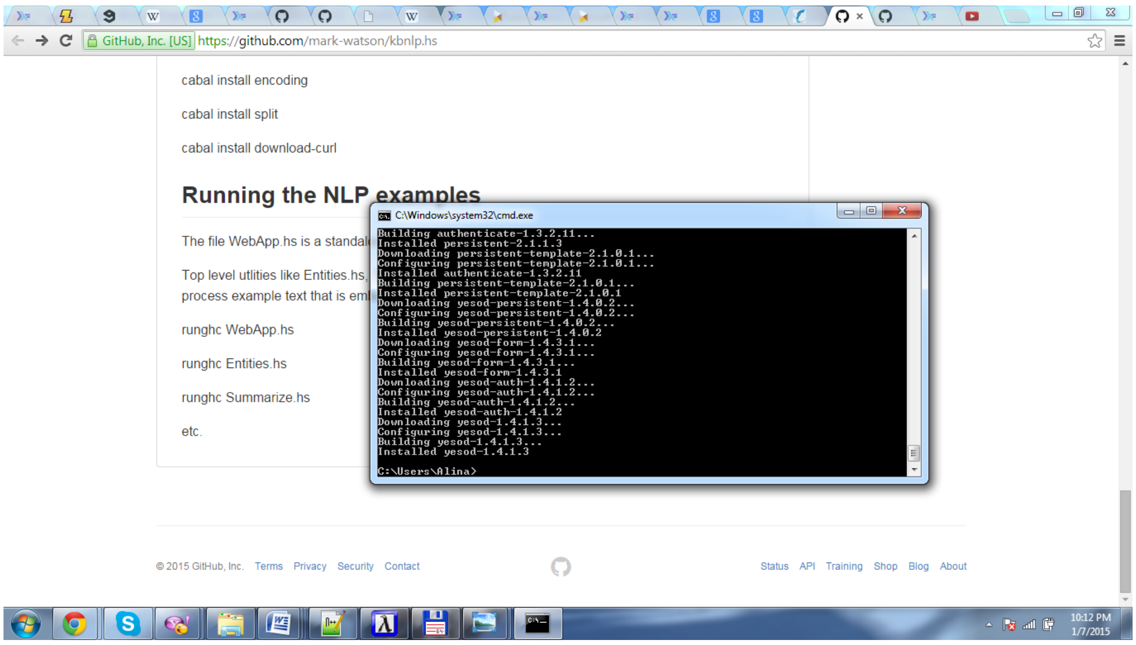
```
let text = pack "The best jokes have no punchline."

print $ tag tagger text
```

4. The output will be an association list of the word to its part of speech:

```
[[ ("The", Tag "at"),

    ("best", Tag "jjt"),

    ("jokes", Tag "nns"),

    ("have", Tag "hv"),

   ("no", Tag "at"),

    ("punchline",Tag "nn"),

    (".",Tag ".") ]]
```

Tutorial 2 KNLP Web application by Mark Watson

>cabal install yesod



https://github.com/mark-watson/kbnlp.hs

cabal install encoding

cabal install split

cabal install download-curl

Running the NLP examples

The file WebApp.hs is a standal

Top level utilities like Entities.hs,
process example text that is em

runghc WebApp.hs

runghc Entities.hs

runghc Summarize.hs

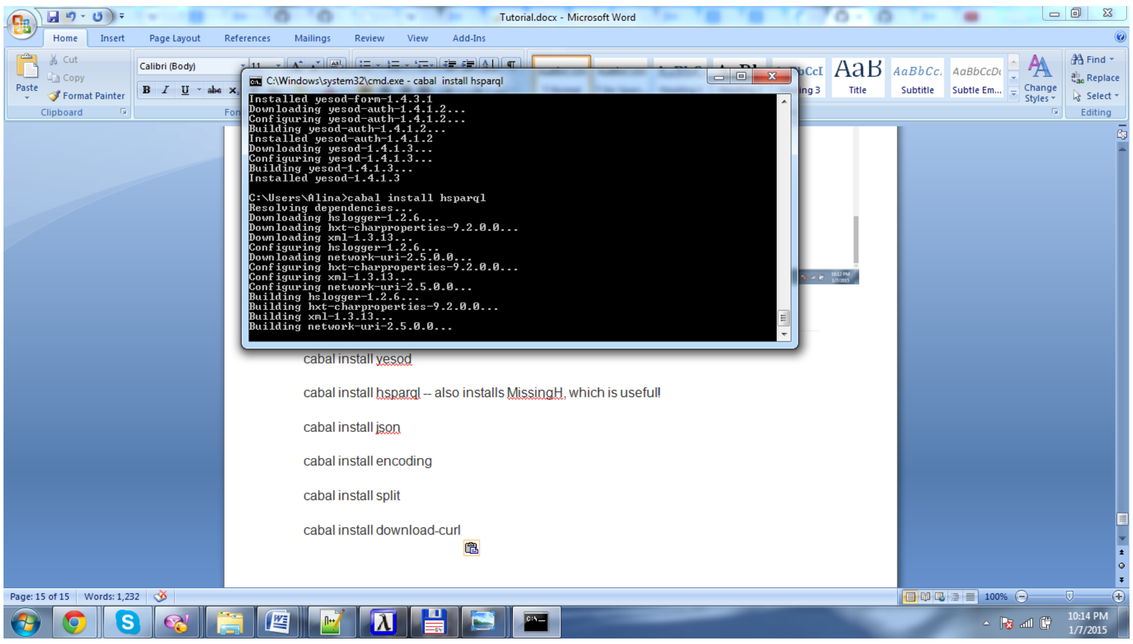
etc.

C:\Windows\system32\cmd.exe

```
Building authenticate-1.3.2.11...
Installed persistent-2.1.1.3
Downloading persistent-template-2.1.0.1...
Configuring persistent-template-2.1.0.1...
Installed authenticate-1.3.2.11
Building persistent-template-2.1.0.1...
Installed persistent-template-2.1.0.1
Downloading yesod-persistent-1.4.0.2...
Configuring yesod-persistent-1.4.0.2...
Building yesod-persistent-1.4.0.2...
Installed yesod-persistent-1.4.0.2
Downloading yesod-form-1.4.3.1...
Configuring yesod-form-1.4.3.1...
Building yesod-form-1.4.3.1...
Installed yesod-form-1.4.3.1
Downloading yesod-auth-1.4.1.2...
Configuring yesod-auth-1.4.1.2...
Building yesod-auth-1.4.1.2...
Installed yesod-auth-1.4.1.2
Downloading yesod-1.4.1.3...
Configuring yesod-1.4.1.3...
Building yesod-1.4.1.3...
Installed yesod-1.4.1.3

C:\Users\Alina>
```

© 2015 GitHub, Inc. Terms Privacy Security Contact
Status API Training Shop Blog About
10:12 PM
1/7/2015

>cabal install hsparql



Tutorial.docx - Microsoft Word

C:\Windows\system32\cmd.exe - cabal install hsparql

```
Installed yesod-form-1.4.3.1
Downloading yesod-auth-1.4.1.2...
Configuring yesod-auth-1.4.1.2...
Building yesod-auth-1.4.1.2...
Installed yesod-auth-1.4.1.2
Downloading yesod-1.4.1.3...
Configuring yesod-1.4.1.3...
Building yesod-1.4.1.3...
Installed yesod-1.4.1.3

C:\Users\Alina>cabal install hsparql
Resolving dependencies...
Downloading hslogger-1.2.6...
Downloading hxt-charproperties-9.2.0.0...
Downloading xml-1.3.13...
Configuring hslogger-1.2.6...
Downloading network-uri-2.5.0.0...
Configuring hxt-charproperties-9.2.0.0...
Configuring xml-1.3.13...
Configuring network-uri-2.5.0.0...
Building hslogger-1.2.6...
Building hxt-charproperties-9.2.0.0...
Building xml-1.3.13...
Building network-uri-2.5.0.0...
```

cabal install yesod

cabal install hsparql -- also installs MissingH, which is usefull

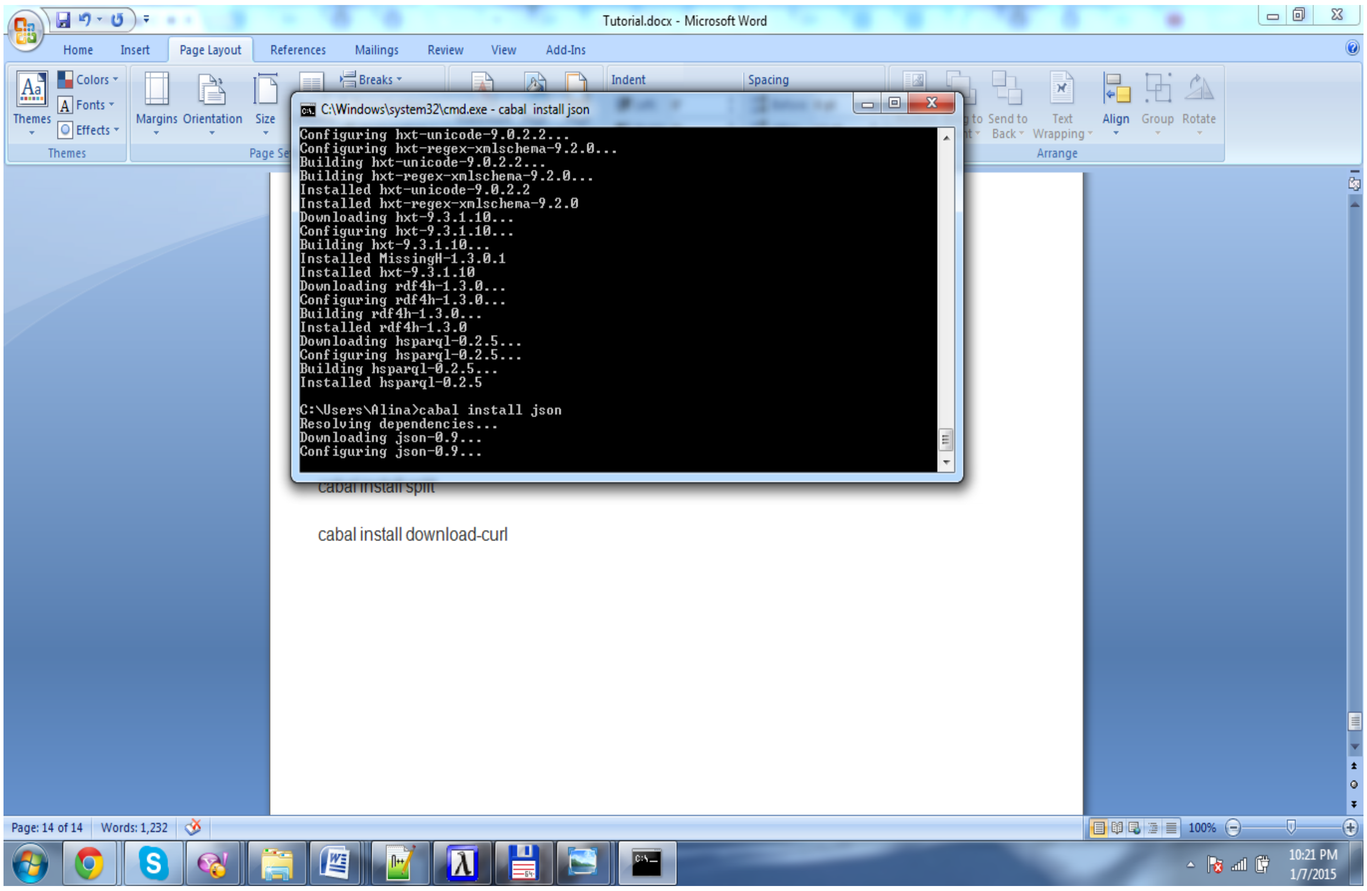
cabal install json

cabal install encoding

cabal install split

cabal install download-curl

>cabal install json

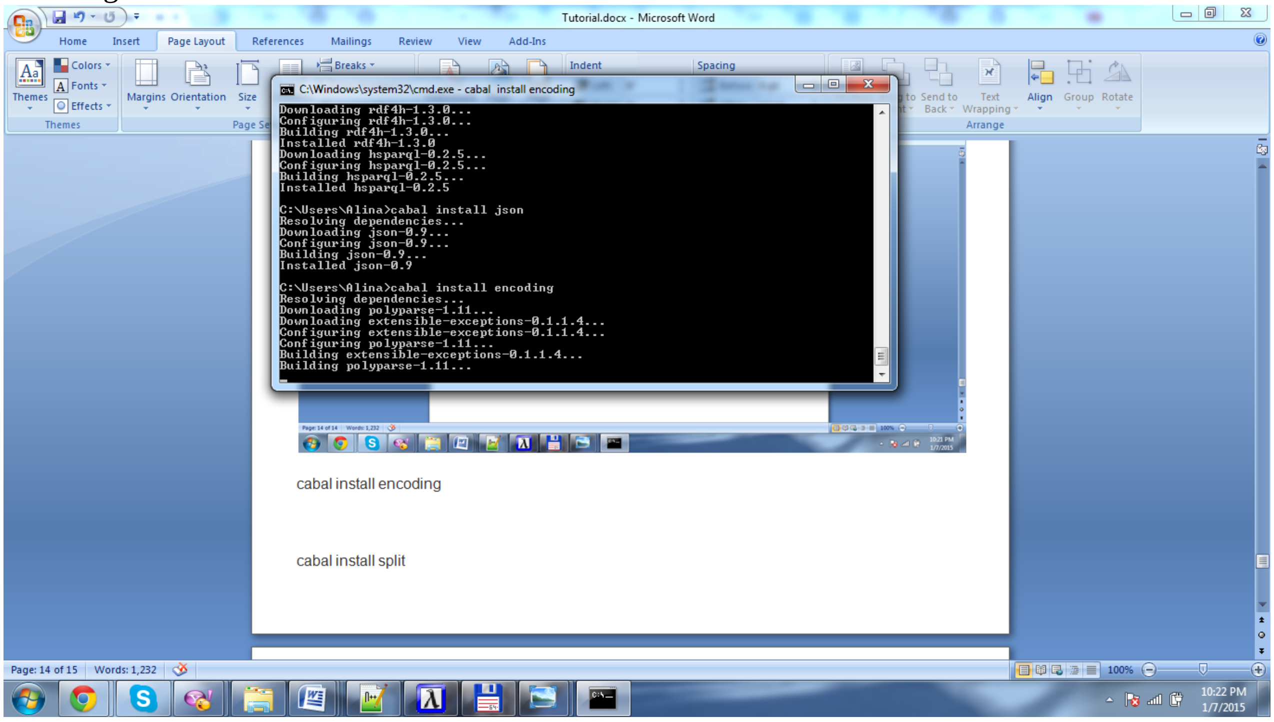

Tutorial.docx - Microsoft Word
C:\Windows\system32\cmd.exe - cabal install json

```
Configuring hxt-unicode-9.0.2.2...
Configuring hxt-regex-xmlschema-9.2.0...
Building hxt-unicode-9.0.2.2...
Building hxt-regex-xmlschema-9.2.0...
Installed hxt-unicode-9.0.2.2
Installed hxt-regex-xmlschema-9.2.0
Downloading hxt-9.3.1.10...
Configuring hxt-9.3.1.10...
Building hxt-9.3.1.10...
Installed MissingH-1.3.0.1
Installed hxt-9.3.1.10
Downloading rdf4h-1.3.0...
Configuring rdf4h-1.3.0...
Building rdf4h-1.3.0...
Installed rdf4h-1.3.0
Downloading hsparql-0.2.5...
Configuring hsparql-0.2.5...
Building hsparql-0.2.5...
Installed hsparql-0.2.5

C:\Users\Alina>cabal install json
Resolving dependencies...
Downloading json-0.9...
Configuring json-0.9...
```

cabal install split

cabal install download-curl

>cabal install
encoding

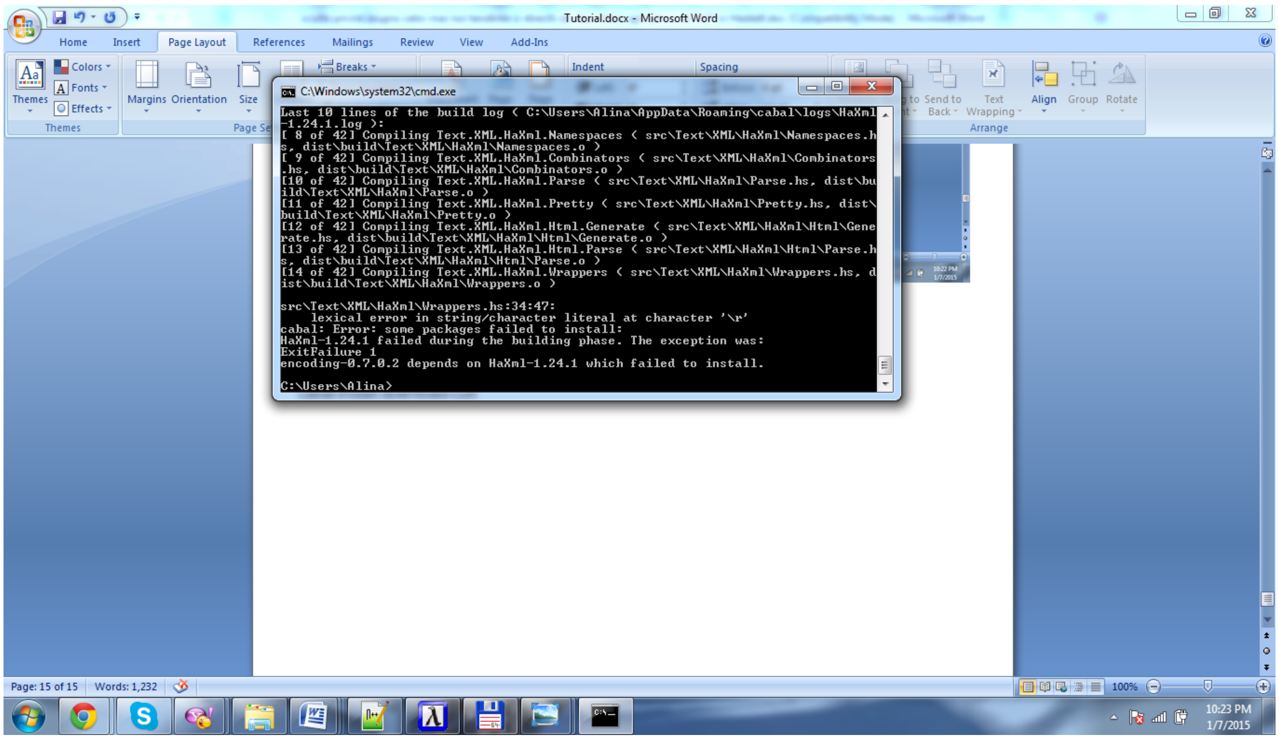


C:\Windows\system32\cmd.exe - cabal install encoding

```
Downloading rdf4h-1.3.0...
Configuring rdf4h-1.3.0...
Building rdf4h-1.3.0...
Installed rdf4h-1.3.0
Downloading hsparql-0.2.5...
Configuring hsparql-0.2.5...
Building hsparql-0.2.5...
Installed hsparql-0.2.5

C:\Users\Alina>cabal install json
Resolving dependencies...
Downloading json-0.9...
Configuring json-0.9...
Building json-0.9...
Installed json-0.9

C:\Users\Alina>cabal install encoding
Resolving dependencies...
Downloading polyparse-1.11...
Downloading extensible-exceptions-0.1.1.4...
Configuring extensible-exceptions-0.1.1.4...
Configuring polyparse-1.11...
Building extensible-exceptions-0.1.1.4...
Building polyparse-1.11...
```

cabal install encoding

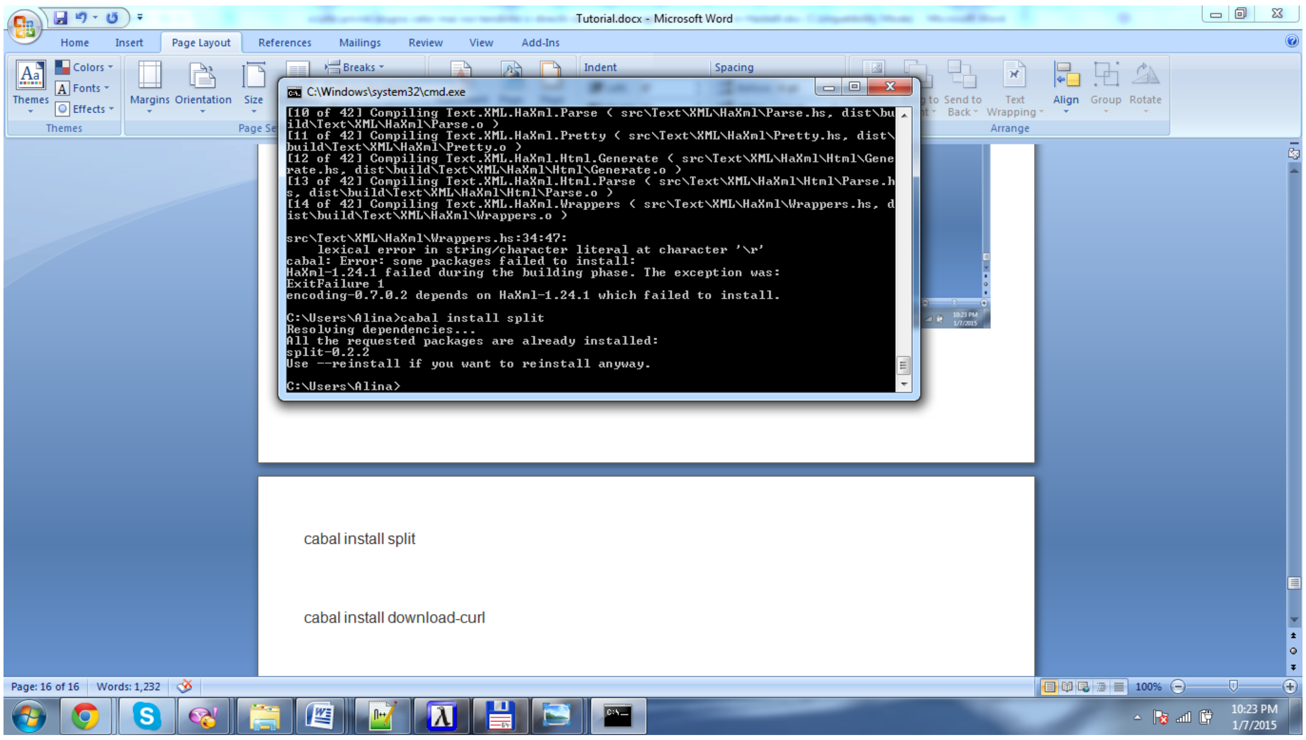
cabal install split



C:\Windows\system32\cmd.exe

```
Last 10 lines of the build log ( C:\Users\Alina\AppData\Roaming\cabal\logs\HaXml
-1.24.1.log ):
[ 8 of 42] Compiling Text.XML.HaXml.Namespaces ( src\Text\XML\HaXml\Namespaces.h
s, dist\build\Text\XML\HaXml\Namespaces.o )
[ 9 of 42] Compiling Text.XML.HaXml.Combinators ( src\Text\XML\HaXml\Combinators
.hs, dist\build\Text\XML\HaXml\Combinators.o )
[10 of 42] Compiling Text.XML.HaXml.Parse ( src\Text\XML\HaXml\Parse.hs, dist\bu
ild\Text\XML\HaXml\Parse.o )
[11 of 42] Compiling Text.XML.HaXml.Pretty ( src\Text\XML\HaXml\Pretty.hs, dist\
build\Text\XML\HaXml\Pretty.o )
[12 of 42] Compiling Text.XML.HaXml.Html.Generate ( src\Text\XML\HaXml\Html\Gene
rate.hs, dist\build\Text\XML\HaXml\Html\Generate.o )
[13 of 42] Compiling Text.XML.HaXml.Html.Parse ( src\Text\XML\HaXml\Html\Parse.h
s, dist\build\Text\XML\HaXml\Html\Parse.o )
[14 of 42] Compiling Text.XML.HaXml.Wrappers ( src\Text\XML\HaXml\Wrappers.hs, d
ist\build\Text\XML\HaXml\Wrappers.o )

src\Text\XML\HaXml\Wrappers.hs:34:47:
    lexical error in string/character literal at character '\r'
cabal: Error: some packages failed to install:
HaXml-1.24.1 failed during the building phase. The exception was:
ExitFailure 1
encoding-0.7.0.2 depends on HaXml-1.24.1 which failed to install.

C:\Users\Alina>
```

>cabal install split



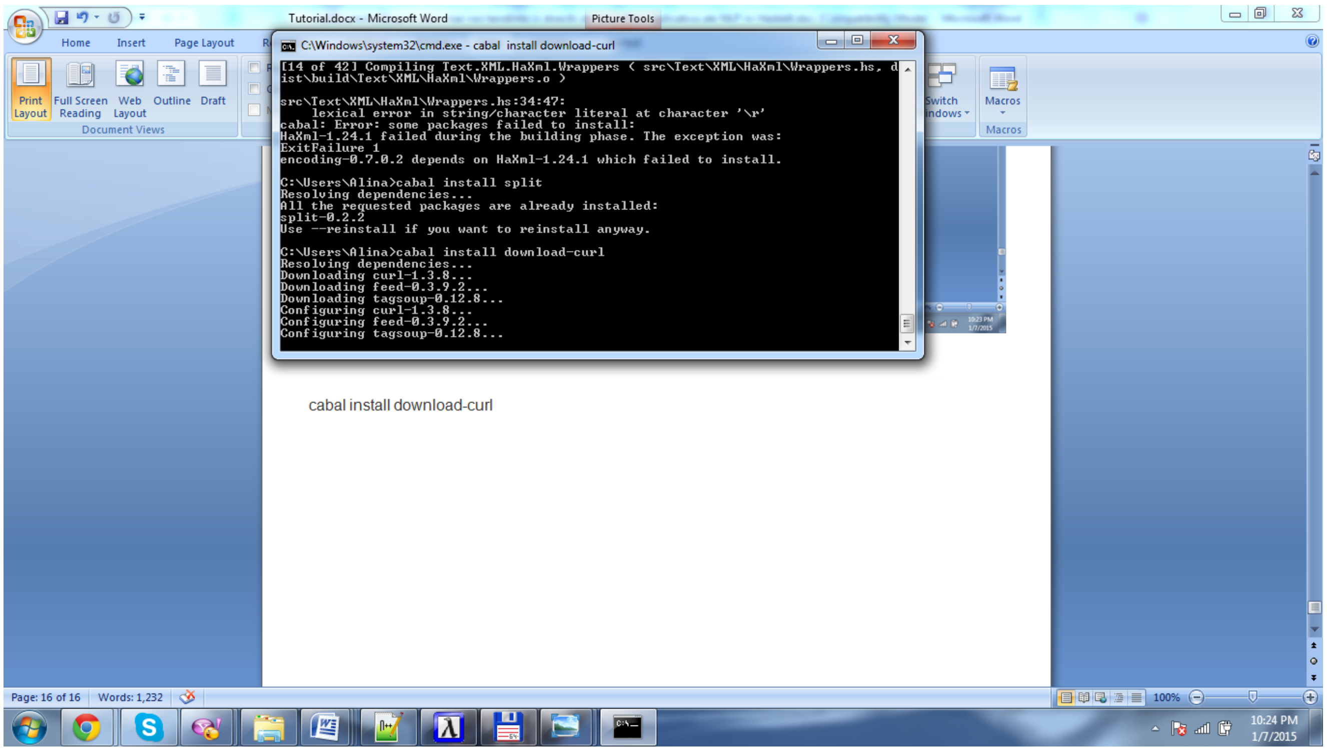
```
[10 of 42] Compiling Text.XML.HaXml.Parse ( src\Text\XML\HaXml\Parse.hs, dist\bu
ild\Text\XML\HaXml\Parse.o )
[11 of 42] Compiling Text.XML.HaXml.Pretty ( src\Text\XML\HaXml\Pretty.hs, dist\
build\Text\XML\HaXml\Pretty.o )
[12 of 42] Compiling Text.XML.HaXml.Html.Generate ( src\Text\XML\HaXml\Html\Gene
rate.hs, dist\build\Text\XML\HaXml\Html\Generate.o )
[13 of 42] Compiling Text.XML.HaXml.Html.Parse ( src\Text\XML\HaXml\Html\Parse.h
s, dist\build\Text\XML\HaXml\Html\Parse.o )
[14 of 42] Compiling Text.XML.HaXml.Wrappers ( src\Text\XML\HaXml\Wrappers.hs, d
ist\build\Text\XML\HaXml\Wrappers.o )

src\Text\XML\HaXml\Wrappers.hs:34:47:
    lexical error in string/character literal at character '\r'
cabal: Error: some packages failed to install:
HaXml-1.24.1 failed during the building phase. The exception was:
ExitFailure 1
encoding-0.7.0.2 depends on HaXml-1.24.1 which failed to install.

C:\Users\Alina>cabal install split
Resolving dependencies...
All the requested packages are already installed:
split-0.2.2
Use --reinstall if you want to reinstall anyway.

C:\Users\Alina>
```

cabal install split

cabal install download-curl

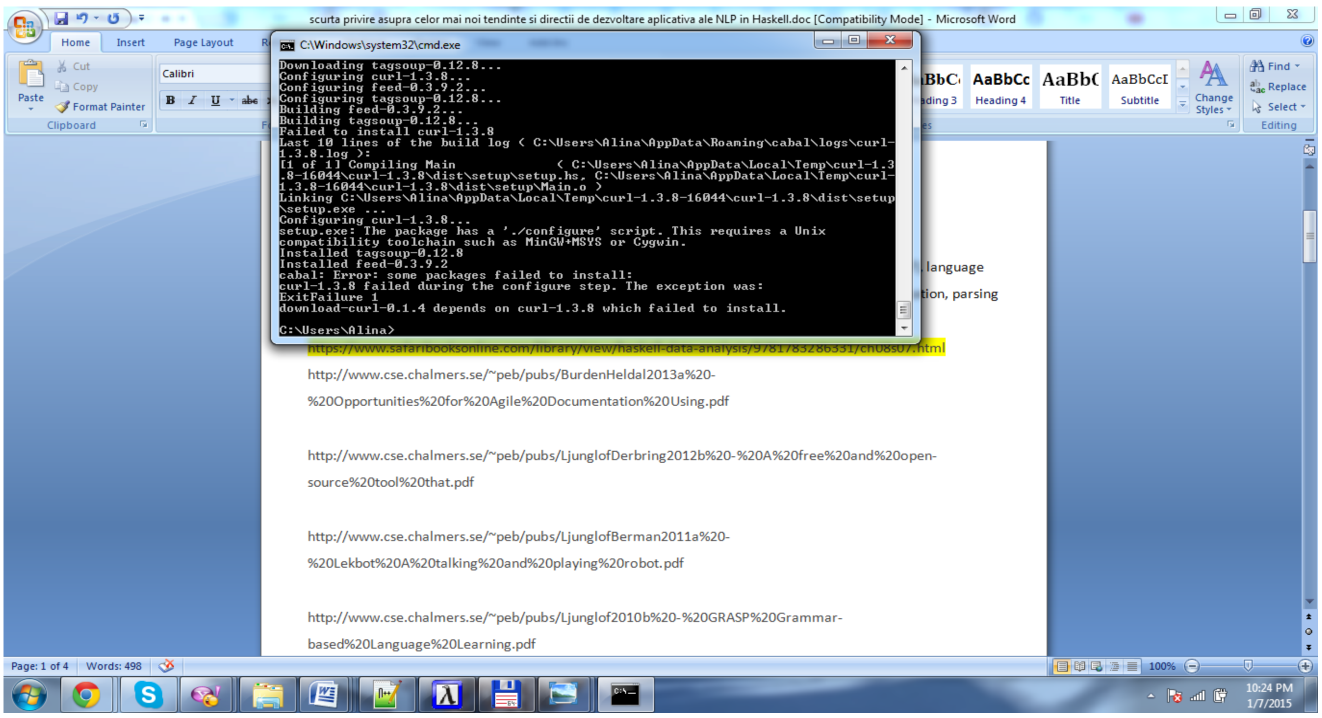
>cabal install download-

curl


Tutorial.docx - Microsoft Word
Picture Tools
C:\Windows\system32\cmd.exe - cabal install download-curl
Page: 16 of 16
Words: 1,232

```
[14 of 42] Compiling Text.XML.HaXml.Wrappers ( src\Text\XML\HaXml\Wrappers.hs, d
ist\build\Text\XML\HaXml\Wrappers.o )

src\Text\XML\HaXml\Wrappers.hs:34:47:
    lexical error in string/character literal at character '\r'
cabal: Error: some packages failed to install:
HaXml-1.24.1 failed during the building phase. The exception was:
ExitFailure 1
encoding-0.7.0.2 depends on HaXml-1.24.1 which failed to install.

C:\Users\Alina>cabal install split
Resolving dependencies...
All the requested packages are already installed:
split-0.2.2
Use --reinstall if you want to reinstall anyway.

C:\Users\Alina>cabal install download-curl
Resolving dependencies...
Downloading curl-1.3.8...
Downloading feed-0.3.9.2...
Downloading tagsoup-0.12.8...
Configuring curl-1.3.8...
Configuring feed-0.3.9.2...
Configuring tagsoup-0.12.8...
```

cabal install download-curl


scurta privire asupra celor mai noi tendinte si directii de dezvoltare aplicativa ale NLP in Haskell.doc [Compatibility Mode] - Microsoft Word
C:\Windows\system32\cmd.exe
Page: 1 of 4
Words: 498

```
Downloading tagsoup-0.12.8...
Configuring curl-1.3.8...
Configuring feed-0.3.9.2...
Configuring tagsoup-0.12.8...
Building feed-0.3.9.2...
Building tagsoup-0.12.8...
Failed to install curl-1.3.8
Last 10 lines of the build log ( C:\Users\Alina\AppData\Roaming\cabal\logs\curl-
1.3.8.log ):
[1 of 1] Compiling Main             ( C:\Users\Alina\AppData\Local\Temp\curl-1.3
.8-16044\curl-1.3.8\dist\setup\setup.hs, C:\Users\Alina\AppData\Local\Temp\curl-
1.3.8-16044\curl-1.3.8\dist\setup\Main.o )
Linking C:\Users\Alina\AppData\Local\Temp\curl-1.3.8-16044\curl-1.3.8\dist\setup
\setup.exe ...
Configuring curl-1.3.8...
setup.exe: The package has a './configure' script. This requires a Unix
compatibility toolchain such as MinGW+MSYS or Cygwin.
Installed tagsoup-0.12.8
Installed feed-0.3.9.2
cabal: Error: some packages failed to install:
curl-1.3.8 failed during the configure step. The exception was:
ExitFailure 1
download-curl-0.1.4 depends on curl-1.3.8 which failed to install.

C:\Users\Alina>
```

https://www.safaribooksonline.com/library/view/haskell-data-analysis/9781783286331/ch08s07.html

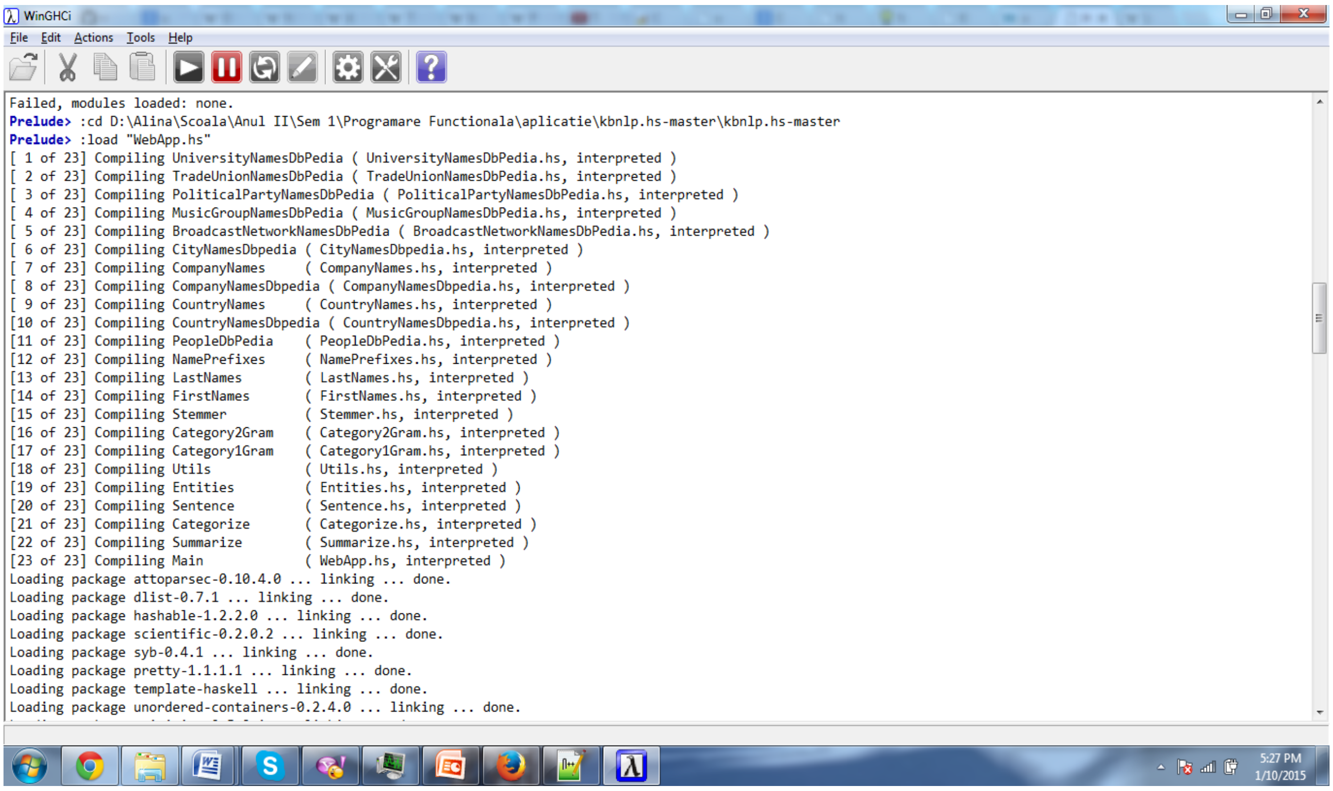
http://www.cse.chalmers.se/~peb/pubs/BurdenHeldal2013a%20-%20Opportunities%20for%20Agile%20Documentation%20Using.pdf

http://www.cse.chalmers.se/~peb/pubs/LjunglofDerbring2012b%20-%20A%20free%20and%20open-source%20tool%20that.pdf

http://www.cse.chalmers.se/~peb/pubs/LjunglofBerman2011a%20-%20Lekbot%20A%20talking%20and%20playing%20robot.pdf

http://www.cse.chalmers.se/~peb/pubs/Ljunglof2010b%20-%20GRASP%20Grammar-based%20Language%20Learning.pdf

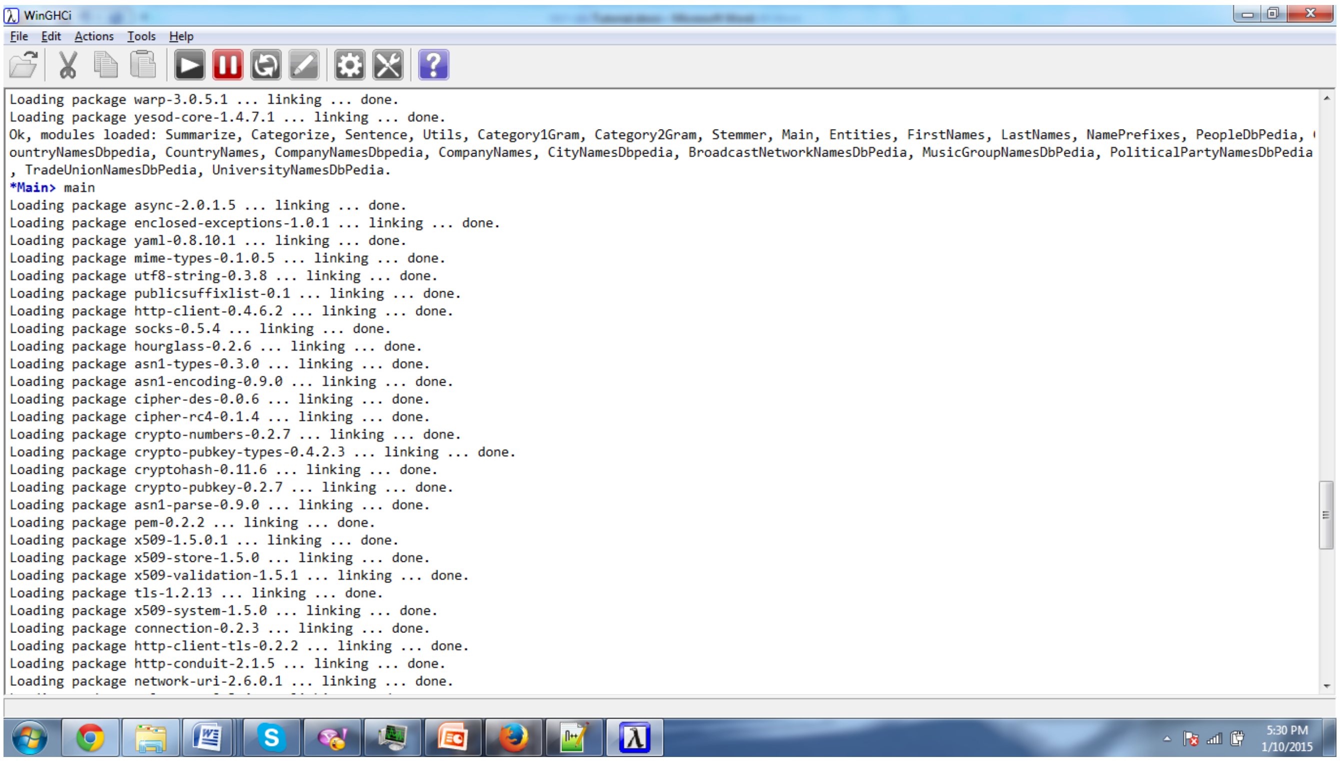
Prelude>:load WebApp.hs



```
Failed, modules loaded: none.
Prelude> :cd D:\Alina\Scoala\Anul II\Sem 1\Programare Functionala\aplicatie\kbnlp.hs-master\kbnlp.hs-master
Prelude> :load "WebApp.hs"
[ 1 of 23] Compiling UniversityNamesDbPedia ( UniversityNamesDbPedia.hs, interpreted )
[ 2 of 23] Compiling TradeUnionNamesDbPedia ( TradeUnionNamesDbPedia.hs, interpreted )
[ 3 of 23] Compiling PoliticalPartyNamesDbPedia ( PoliticalPartyNamesDbPedia.hs, interpreted )
[ 4 of 23] Compiling MusicGroupNamesDbPedia ( MusicGroupNamesDbPedia.hs, interpreted )
[ 5 of 23] Compiling BroadcastNetworkNamesDbPedia ( BroadcastNetworkNamesDbPedia.hs, interpreted )
[ 6 of 23] Compiling CityNamesDbpedia ( CityNamesDbpedia.hs, interpreted )
[ 7 of 23] Compiling CompanyNames     ( CompanyNames.hs, interpreted )
[ 8 of 23] Compiling CompanyNamesDbpedia ( CompanyNamesDbpedia.hs, interpreted )
[ 9 of 23] Compiling CountryNames     ( CountryNames.hs, interpreted )
[10 of 23] Compiling CountryNamesDbpedia ( CountryNamesDbpedia.hs, interpreted )
[11 of 23] Compiling PeopleDbPedia    ( PeopleDbPedia.hs, interpreted )
[12 of 23] Compiling NamePrefixes     ( NamePrefixes.hs, interpreted )
[13 of 23] Compiling LastNames        ( LastNames.hs, interpreted )
[14 of 23] Compiling FirstNames       ( FirstNames.hs, interpreted )
[15 of 23] Compiling Stemmer          ( Stemmer.hs, interpreted )
[16 of 23] Compiling Category2Gram    ( Category2Gram.hs, interpreted )
[17 of 23] Compiling Category1Gram    ( Category1Gram.hs, interpreted )
[18 of 23] Compiling Utils            ( Utils.hs, interpreted )
[19 of 23] Compiling Entities         ( Entities.hs, interpreted )
[20 of 23] Compiling Sentence         ( Sentence.hs, interpreted )
[21 of 23] Compiling Categorize       ( Categorize.hs, interpreted )
[22 of 23] Compiling Summarize        ( Summarize.hs, interpreted )
[23 of 23] Compiling Main             ( WebApp.hs, interpreted )
Loading package attoparsec-0.10.4.0 ... linking ... done.
Loading package dlist-0.7.1 ... linking ... done.
Loading package hashable-1.2.2.0 ... linking ... done.
Loading package scientific-0.2.0.2 ... linking ... done.
Loading package syb-0.4.1 ... linking ... done.
Loading package pretty-1.1.1.1 ... linking ... done.
Loading package template-haskell ... linking ... done.
Loading package unordered-containers-0.2.4.0 ... linking ... done.
```

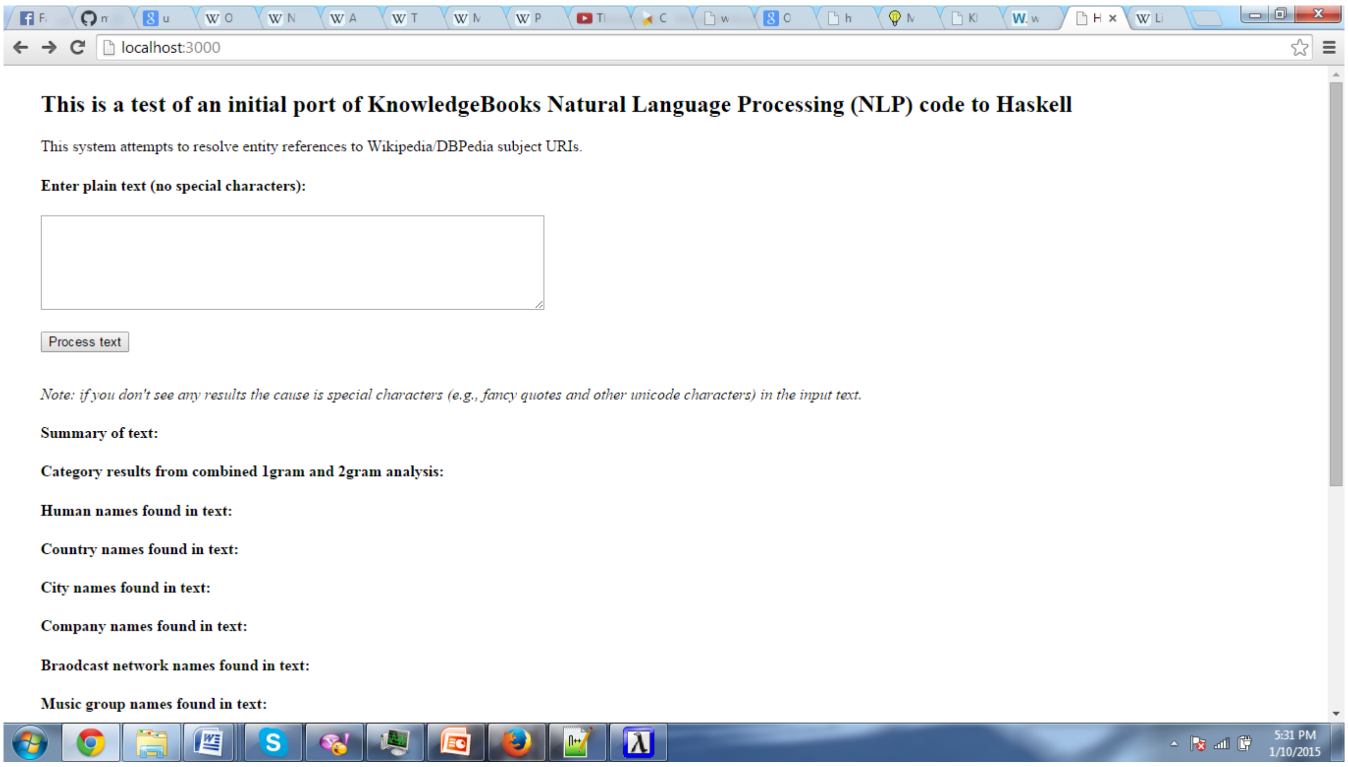
start yesod server

Main>main



```
Loading package warp-3.0.5.1 ... linking ... done.
Loading package yesod-core-1.4.7.1 ... linking ... done.
Ok, modules loaded: Summarize, Categorize, Sentence, Utils, Category1Gram, Category2Gram, Stemmer, Main, Entities, FirstNames, LastNames, NamePrefixes, PeopleDbPedia, C
ountryNamesDbpedia, CountryNames, CompanyNamesDbpedia, CompanyNames, CityNamesDbpedia, BroadcastNetworkNamesDbPedia, MusicGroupNamesDbPedia, PoliticalPartyNamesDbPedia
, TradeUnionNamesDbPedia, UniversityNamesDbPedia.
*Main> main
Loading package async-2.0.1.5 ... linking ... done.
Loading package enclosed-exceptions-1.0.1 ... linking ... done.
Loading package yaml-0.8.10.1 ... linking ... done.
Loading package mime-types-0.1.0.5 ... linking ... done.
Loading package utf8-string-0.3.8 ... linking ... done.
Loading package publicsuffixlist-0.1 ... linking ... done.
Loading package http-client-0.4.6.2 ... linking ... done.
Loading package socks-0.5.4 ... linking ... done.
Loading package hourglass-0.2.6 ... linking ... done.
Loading package asn1-types-0.3.0 ... linking ... done.
Loading package asn1-encoding-0.9.0 ... linking ... done.
Loading package cipher-des-0.0.6 ... linking ... done.
Loading package cipher-rc4-0.1.4 ... linking ... done.
Loading package crypto-numbers-0.2.7 ... linking ... done.
Loading package crypto-pubkey-types-0.4.2.3 ... linking ... done.
Loading package cryptohash-0.11.6 ... linking ... done.
Loading package crypto-pubkey-0.2.7 ... linking ... done.
Loading package asn1-parse-0.9.0 ... linking ... done.
Loading package pem-0.2.2 ... linking ... done.
Loading package x509-1.5.0.1 ... linking ... done.
Loading package x509-store-1.5.0 ... linking ... done.
Loading package x509-validation-1.5.1 ... linking ... done.
Loading package tls-1.2.13 ... linking ... done.
Loading package x509-system-1.5.0 ... linking ... done.
Loading package connection-0.2.3 ... linking ... done.
Loading package http-client-tls-0.2.2 ... linking ... done.
Loading package http-conduit-2.1.5 ... linking ... done.
Loading package network-uri-2.6.0.1 ... linking ... done.
```

localhost:3000



This is a test of an initial port of KnowledgeBooks Natural Language Processing (NLP) code to Haskell

This system attempts to resolve entity references to Wikipedia/DBPedia subject URIs.

Enter plain text (no special characters):

Note: if you don't see any results the cause is special characters (e.g., fancy quotes and other unicode characters) in the input text.

Summary of text:

Category results from combined 1gram and 2gram analysis:

Human names found in text:

Country names found in text:

City names found in text:

Company names found in text:

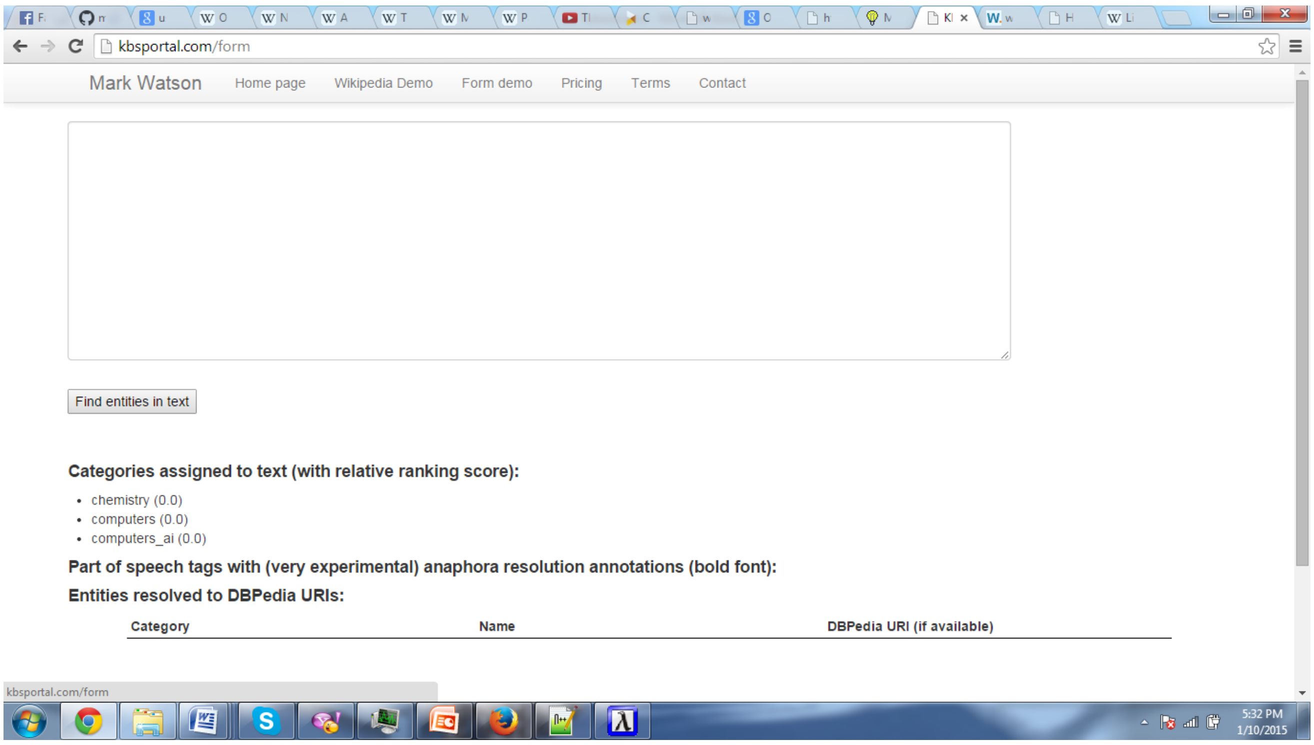
Braodcast network names found in text:

Music group names found in text:

author's website:

http://markwatson.com/

demo



Categories assigned to text (with relative ranking score):

- chemistry (0.0)
- computers (0.0)
- computers_ai (0.0)

Part of speech tags with (very experimental) anaphora resolution annotations (bold font):

Entities resolved to DBPedia URIs:

Category	Name	DBPedia URI (if available)

documentation for the app:

http://kbsportal.com/demo

Bibliography:

- *Natural Language Processing for the Working Programmer*

  2010, 2011 Daniël de Kok, Harm Brouwer

- Jack Copeland, (18 June 2012). "Alan Turing: The codebreaker who saved 'millions of lives'". BBC News Technology. Retrieved 26 October 2014

- Dipanjan Das, André F.T. Martins, “A Survey on Automatic Text Summarization”, Carnegie Mellon University, November 2007

- http://translation-software-review.toptenreviews.com/

- Benjamin Bengfort, “An Introduction to Named Entity Recognition in Natural Language Processing - Part 1”, 17 April 2013, www.datacommunnitydc.org

- Claire Gardent, “*Natural Language Generation*”, CNRS/LORIA, Nancy, http://www.loria.fr/~gardent/teaching/nlg-bkk06.pdf

  Steven Bird, Ewan Klein, and Edward Loper, “*Natural Language Processing*”, O’Reilly Media, 2009

- www.semantria.com “*What is Sentiment Analysis?”*

- http://homepages.inf.ed.ac.uk/wadler/realworld/natlangproc.html
- http://en.wikipedia.org/wiki/LOLITA

- http://hackage.haskell.org/package/chatter

- http://en.wikipedia.org/wiki/Brown_Corpus#Part-of-speech_tags_used.

- *Grammar-Based Automated Music Composition in Haskell*

  Donya Quick and Paul Hudak, Yale University, 2013

- *The Haskell School of Music — From Signals to Symphonies —*

  Paul Hudak, Yale University, Department of Computer Science, January 2014

- http://www.ibm.com/developerworks/library/j-ft20/
- http://www.smashingmagazine.com/2014/07/02/dont-be-scared-of-functional-programming/

- http://www.kurzweilai.net/psychopathic-killers-computerized-text-analysis-uncovers-the-word-patterns-of-a-predator
- http://haskell.cs.yale.edu/