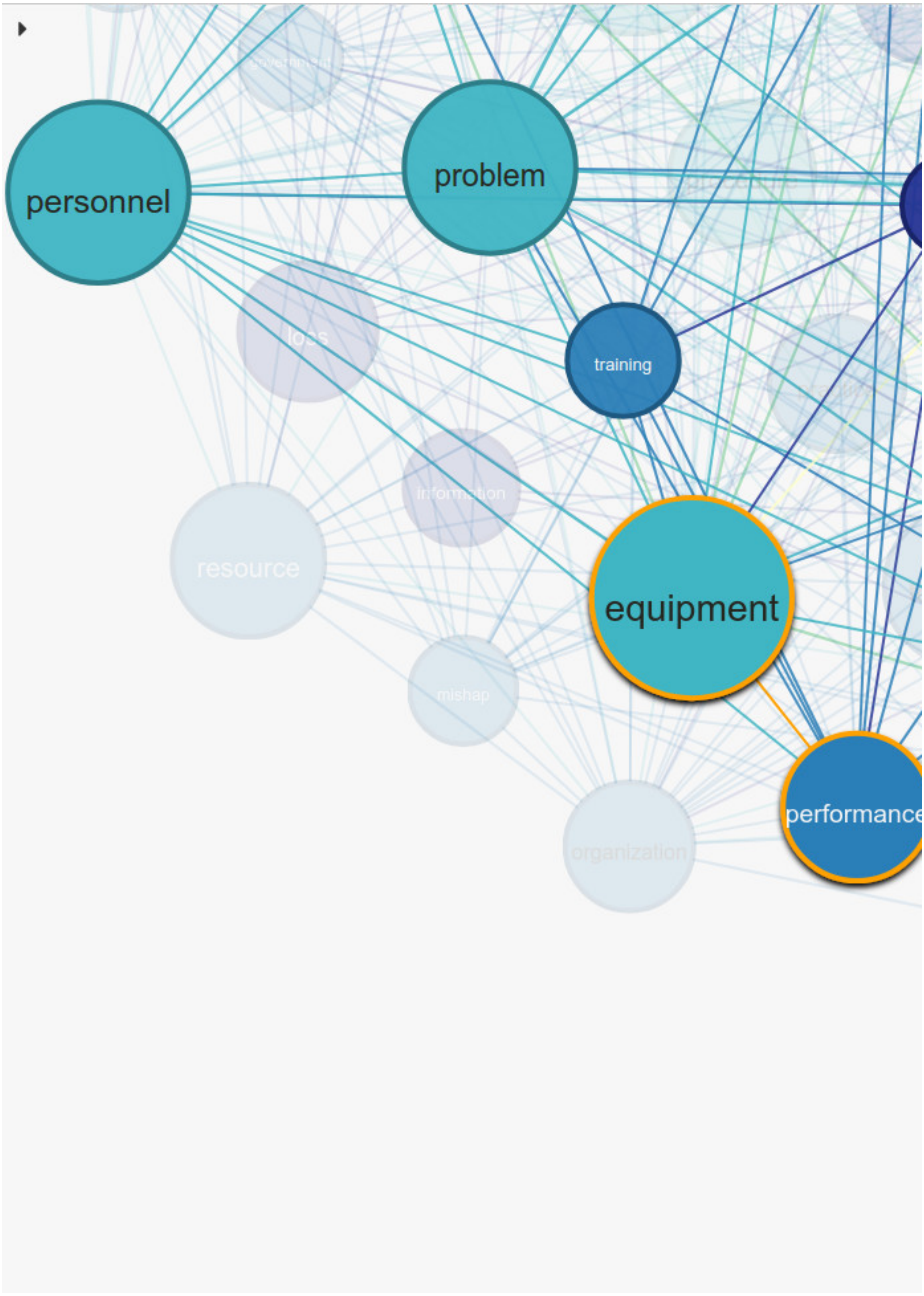
Companies of any size have to manage and access huge amounts of data providing advanced services for their end-users or to handle their internal processes. The greater part of this data is usually stored in the form of text. Processing and analyzing this huge source of knowledge represents a competitive advantage, but often, even providing simple and effective access to it is a complex task, due to the unstructured nature of the textual data. This blog post will focus on a specific use case: provide effective access to a huge set of documents - later referred as a corpus - identifying **main concepts** , organizing **indexes** and providing an **adequate visualization**.

Keyword extraction is the identification and selection of words or small phrases that best describe a document. Such keywords may constitute useful entries for building indexes for a corpus, can be used to classify text, or can serve as a simple summary for a given document. Moreover, a system for automatic identification of important terms in text can be used for the problem of terminology extraction, and construction of domain-specific dictionaries.

The following paragraphs describe a method for keyword extraction that leverages a graph model representing the relationships between tags or concepts in the documents. The solution starts from a graph-based unsupervised technique called TextRank [1]. Thereafter, the quality of extracted keywords is greatly improved using a **typed dependency graph** that is used to filter out meaningless phrases, or to extend keywords with adjectives and nouns to better describe the text. It is worth noting here that the proposed approach is completely un-supervised, nevertheless, the final results can be compared with supervised approaches.

In order to test the method and provide a concrete application of the algorithm, the NASA Lessons Learned public dataset has been used. The proposed visualization will show one of the possible ways for navigating and accessing the data using the information automatically extracted from the corpus. The following image is a first example of the results achieved.


personnel
problem
training
loss
information
resource
equipment
mishap
organization
performance

NLP TextRank Image1

GraphAware Knowledge Platform

The GraphAware Knowledge Platform, later referred to as the Platform, is an adaptive platform for centralized, deep analysis of structured and unstructured data, composing the enterprise knowledge base. It represents the logical evolution of the **GraphAware NLP Framework** which forms the core of a more complex environment for managing several data sources and offers an even higher level set of features. The Platform also provides a Graphical User Interface, the **GraphAware Knowledge Studio** and a set of APIs, the **GraphAware Knowledge API**. The former allows configuration and management of the platform, as well as navigating the data. The latter provides a secure REST interface for easy integration of features into existing infrastructures. The NLP Framework has been extended to support unsupervised keyword extraction. As stated in previous blog posts, it integrates NLP processing capabilities available in several software packages like Stanford NLP and OpenNLP, existing data sources, such as ConceptNet 5 and WordNet, and GraphAware knowledge about search, graphs, and recommendation engines. GraphAware NLP is implemented as a plugin for Neo4j and an external frontend for interacting with a Spark Cluster. It provides a set of tools, by means of procedures, background process, and APIs, which together provide a Domain Specific Language for Natural Language Processing on top of Cypher. The available interesting features are:

- Information Extraction (IE) - processing textual information for extracting main components and relationships
- Extracting sentiment
- Enriching basic data with ontologies and concepts (ConceptNet 5)
- Computing similarities between text elements in a corpus using base data and ontology information
- Enriching knowledge using external sources (Alchemy)
- Providing basic search capabilities
- Providing complex search capabilities leveraging enriched knowledge such as ontology, sentiment, and similarity
- Providing recommendations based on a combination of content/ontology-based recommendations, social tags, and collaborative filtering
- Unsupervised corpus clustering using LDA
- Semi-supervised corpus clustering using Label Propagation
- Word2Vec computation and importing

More about GraphAware NLP can be found in the NLP section of the GraphAware blog. The Knowledge Studio has been used as a valuable tool for visualizing and accessing data leveraging the keywords automatically inferred using the implemented feature.

TextRank

The TextRank algorithm, introduced in [1], is a relatively simple, unsupervised method of text summarization directly applicable to the topic extraction task. Its objective is to retrieve keywords and construct key phrases that are most descriptive of a given document by building a graph of word co-occurrences and ranking the importance of individual words using the PageRank algorithm.

The key steps are as follows:

- **Pre-select relevant words from the NLP annotated text**. Each document is tokenized and annotated using the GraphAware NLP Framework (these processed words are basic lexical units, also denoted as tags from now on). A stopword list (configurable) and a syntactic filter are applied to refine the selection to the most relevant lexical units. The syntactic filter selects only nouns and adjectives, following [1] the observation that even human annotators tend to use nouns rather than verb phrases to summarize documents.
- **Create a graph of tag co-occurrences**. Filtered tags are ordered based on their position in the document and co-occurrence relationships are built between adjacent tags, following the natural word flow in the text. This introduces the relations between syntactic elements of the document into the graph. By default, only tags that appear next to each other can have a co-occurrence relationship. For example, in the sentence "Pieter eats fish.", no co-occurrence edge is created because eats is a verb that didn't pass the syntactic filter. However, if the co-occurrence window (default is 2) is changed to 3, Pieter and fish will become connected. Finally, each co-occurrence edge is assigned a weight property indicating the number of times the two tags co-occurred within the given document.
- **Run undirected weighted PageRank**. The undirected PageRank algorithm is run on weighted co-occurrence relationships to rate nodes (tags) based on their importance in the graph. Experiments with unweighted PageRank showed up that weights are useful in bringing forward important keywords (such as space shuttle, which wouldn't be otherwise reconstructed in some NASA Lessons Learned documents).
- **Save top 1/3 of tags as keywords and identify key phrases**. The tags are ordered based on the PageRank ratings, and then the top 1/3 (configurable) are taken as final keywords. If some of these selected tags are adjacent, they are collapsed into a key phrase.

The identified keywords and key phrases are saved to the graph database via a **DESCRIBES** relationship between a keyword node **Keyword** and the **AnnotatedText** node. Label names are configurable.

First Results with Text Rank

The GraphAware TextRank procedure has many useful parameters allowing user-customization, but only one is mandatory: the AnnotatedText node. No other argument needs to be specified - the TextRank performs very well out-of-the-box:

```
// Run TextRank
MATCH (a:AnnotatedText)
WITH
CALL ga.nlp.ml.textRank({annotatedText: a}) YIELD result
RETURN count(*) AS n_processedDocuments
```

To see the reconstructed key phrases (multi-word keywords), this is the Cypher query:

```
// Inspect multi-word keywords
MATCH (k:Keyword)-[:DESCRIBES]->(a:AnnotatedText)
WHERE size(split(k.value, " "))>1
RETURN k.value AS Keyphrase, count(*) AS n_lessons
ORDER BY n_lessons DESC
```

And indeed, the results are very promising: in total, the algorithm produces around 7200 key phrases (on an average 4 to 5 key phrases per lesson), out of which 775 occur in two and more lessons and 300 in three and more lessons. The table below presents the top 20 key phrases:

Key phrase	**# of lessons**
space shuttle	38
International Space Station	37
flight hardware	37
Kennedy Space Center	24
project management	21
flight software	15
launch vehicle	14
project manager	14
flight system	14
system design	14
Solid Rocket Motor	13
flight project	13
shuttle program	13
software development	12
mission operation	12
system engineering	12
space flight	11
NASA project	11
ground processing	11
risk management	10

–

The key phrases in the list appear very relevant to the NASA corpus. However, it is also important to note here that out of those hundreds of key phrases, there are also clear misidentifications - phrases like *overall historical* or *lower operation* don't seem very helpful. One of the options that could help is to improve the stopwords list, however, there's also a more sophisticated approach discussed in the section below.

Algorithm Enhancement

The first results obtained so far with the TextRank appeared to be really promising, but the quality can be improved using more insights about the text. The basic algorithm has been modified in order to leverage the typed dependency graph provided by Stanford NLP and integrated in the NLP Framework.

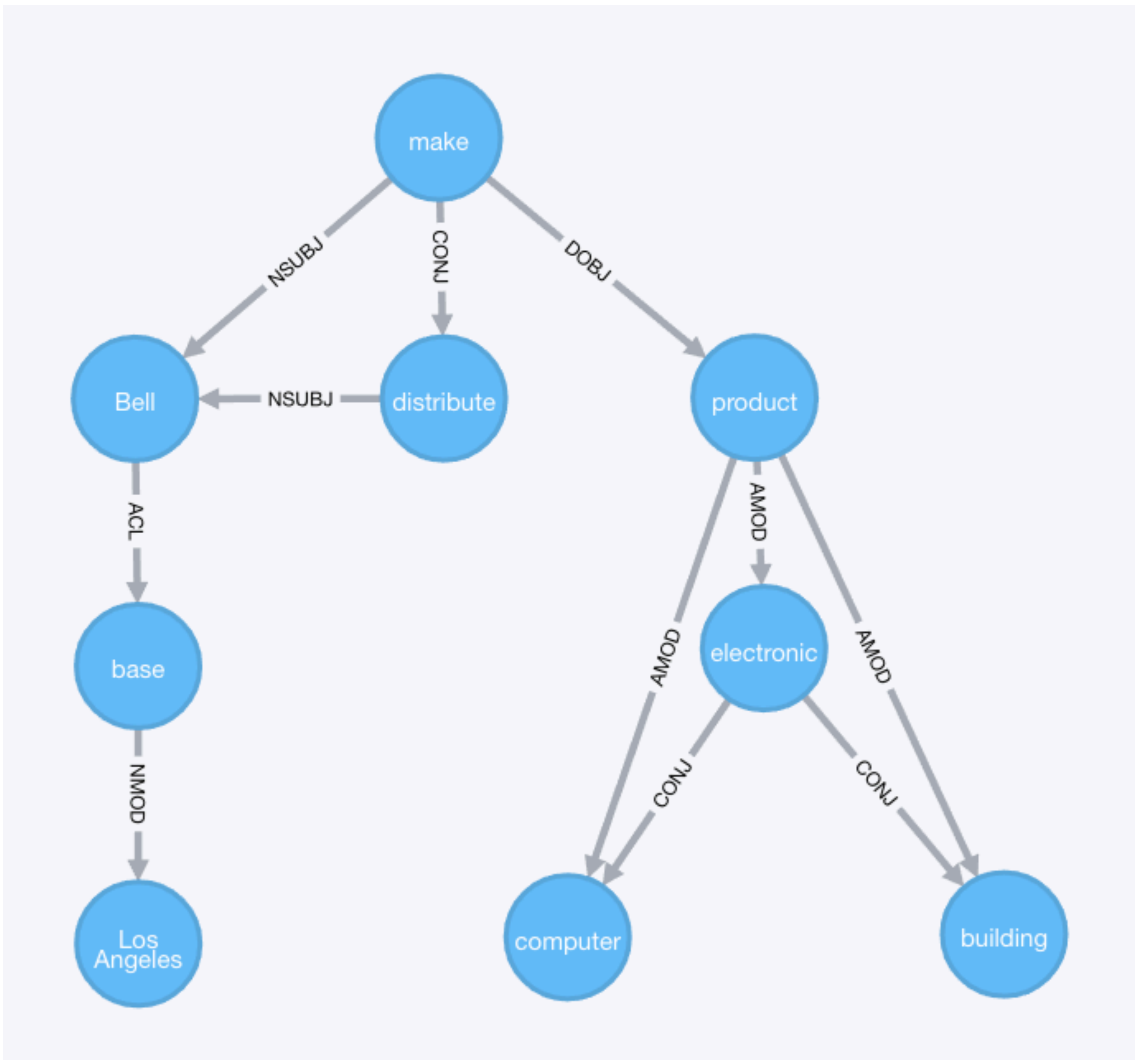
Stanford Typed Dependencies

Stanford typed dependencies provide a simple way to visualize and analyze grammatical relationships between words in a sentence. They are designed such that they can be easily understood and used even by non-linguists to leverage textual relations for information extraction tasks. Stanford dependencies are represented as triplets: name of the relation, governor and dependent. [2]

Here is an example sentence:

```
“Bell, based in Los Angeles, makes and distributes electronic, computer and building
products.”
```

The results of the text annotation and analysis of dependencies are showed in the next graph :



NLP TextRank Image2

In order to improve the quality of Automatic Keyword Extraction, the algorithm considers the following typed dependencies:

amod: adjectival modifier

An adjectival modifier of an NP is any adjectival phrase that serves to modify the meaning of the NP.

```
“Sam eats red meat” -> amod(meat, red)
“Sam took out a 3 million dollar loan” -> amod(loan, dollar)
“Sam took out a $ 3 million loan” -> amod(loan, $)
```

nn: noun compound modifier

A noun compound modifier of an NP is any noun that serves to modify the head noun. (Note that in the current system for dependency extraction, all nouns modify the rightmost noun of the NP – there is no intelligent noun compound analysis. This is likely to be fixed once the Penn Treebank represents the branching structure of NPs.) [2]

```
“Oil price futures” -> nn(futures, oil), nn(futures, price)
```

New TextRank with Dependencies

As discussed above, the TextRank algorithm in its basic form suffers from certain shortcomings, which could, however, be overcome by the use of typed dependencies. Let’s examine the following key phrase:

personal protective

- both words are adjectives => clearly a noun is missing
- inspecting the relevant NASA lessons: the missing word is equipment. This happened because this word was ranked just below the threshold of ⅓ of top words, in some other lessons it passed the threshold but the word *personal* didn’t. As a result, documents discussing the same topic - *personal protective* equipment - were not assigned any common key phrase!
- AMOD dependencies are between: *personal - equipment* and *protective - equipment*

Typed dependencies can be used not only to remove key phrases that have no mutual relationship of type COMPOUND or AMOD (such as many space, one FAA etc.), but also to complete existing key phrases with missing tag(s) due to imperfection of the ranking process.

The improved TextRank algorithm therefore introduces two new principles:

- **All tags** in a key phrase candidate must be related through COMPOUND or AMOD dependency
- If they are not, but are **related to another neighbouring tag** that was not ranked high enough to be included in the key phrase by the original TextRank, **that tag is added** (this dependency enrichment is also performed for single-word keywords)

The latter principle takes care of handling the previously mentioned shortcomings and also adds a higher level of detail. For example, a NASA lesson where *space station* was an identified phrase. It is obviously a topic highly relevant to many documents in the corpus. However, a closer look at that lesson reveals that it actually talked about a particular space station, the International Space Station. Since the tag international was not ranked high enough in the graph of co-occurrences, it was not merged with *space station*. And here the typed dependencies come to rescue once again: there are COMPOUND dependencies between (international, station) and (space, station).

Although being able to reconstruct the whole phrase international space station is certainly desirable, it could be useful to preserve the knowledge that this is just a specific case of a more general space station group. Similarly with these other examples:

```
coaxial rf cable failure
onboard spacecraft tape recorder
launch space flight hardware
space shuttle logistics program
Agency-wide generic full cost accounting system
```

These results resemble a lot key phrases that many human annotators would use to describe given documents, but it is also obvious that they can become too detailed: could be also interesting to know that these lessons discuss RF cables, tape recorders, flight hardware, Space Shuttles and accounting systems in general.

To deal with the issue of excessive level of detail introduced by typed dependencies, a simple post-processing procedure was designed and can be invoked the following way:

```
CALL ga.nlp.ml.textRank.postprocess({keywordLabel: "Keyword"}) YIELD result
RETURN count(*)
```

Each key phrase of n number of tags is checked against all key phrases from all documents with 1 < m < n number of tags. If the former contains the latter key phrase, then a DESCRIBES relationship is created from the m-keyphrase to all annotated texts of the n-keyphrase. Additionally, a HAS_SUBGROUP relationship is created between both key phrases to clearly mark a parent and child. This provides an easy way to set a preference for the most detailed key phrases only:

```
// Get lesson count per key phrase for the most detailed phrases only
MATCH (k:Keyword)-[:DESCRIBES]->(a:AnnotatedText)
WHERE k.numTerms > 1 AND NOT (k)-[:HAS_SUBGROUP]->(:Keyword)-[:DESCRIBES]->(a)
RETURN k.value as Keyphrase, count(*) AS n_lessons
ORDER BY n_lessons DESC
LIMIT 20
```

With all these upgrades put together, the list of key phrases contains almost 14000 elements, out of which 2570 occur in two and more lessons and 1193 in three and more lessons. The table below presents the top 20 key phrases (no requirement on HAS_SUBGROUP relationships):

Key phrase	**# of lessons**
flight hardware	81
space shuttle	60
International Space Station	48
project management	47

Key phrase	**# of lessons**
shuttle program	44
ground system	38
flight system	33
launch vehicle	32
risk management	29
control system	29
flight project	29
system design	28
Kennedy Space Center	27
failure mode	26
project manager	26
space flight	25
configuration control	25
space program	25
constellation program	23
system engineering	22

–

Comparing this table to the previous one (TextRank without dependencies), it is possible to notice a clear improvement, for example: space shuttle is identified as an important key phrase in 60 lessons (it was 38), flight hardware in 81 (it was 37) etc. And there are many new highly relevant key phrases: stainless steel tank, cryostat helium tank, pneumatic valve, gas transfer valve etc.

And similarly top 20 single-word keywords (about 2,400 in total, out of which 1154 are present in 2 and more lessons):

Key phrase	**# of lessons**
cause	66
result	46
design	46
failure	41
lack	36
test	36
launch	34
lead	33
project	32
NASA	31
increase	30
schedule	29
program	29
equipment	29
prior	28

Key phrase	# of lessons
mission	27
system	26
determine	25
need	25
problem	25

–

As an example, take this NASA lesson:

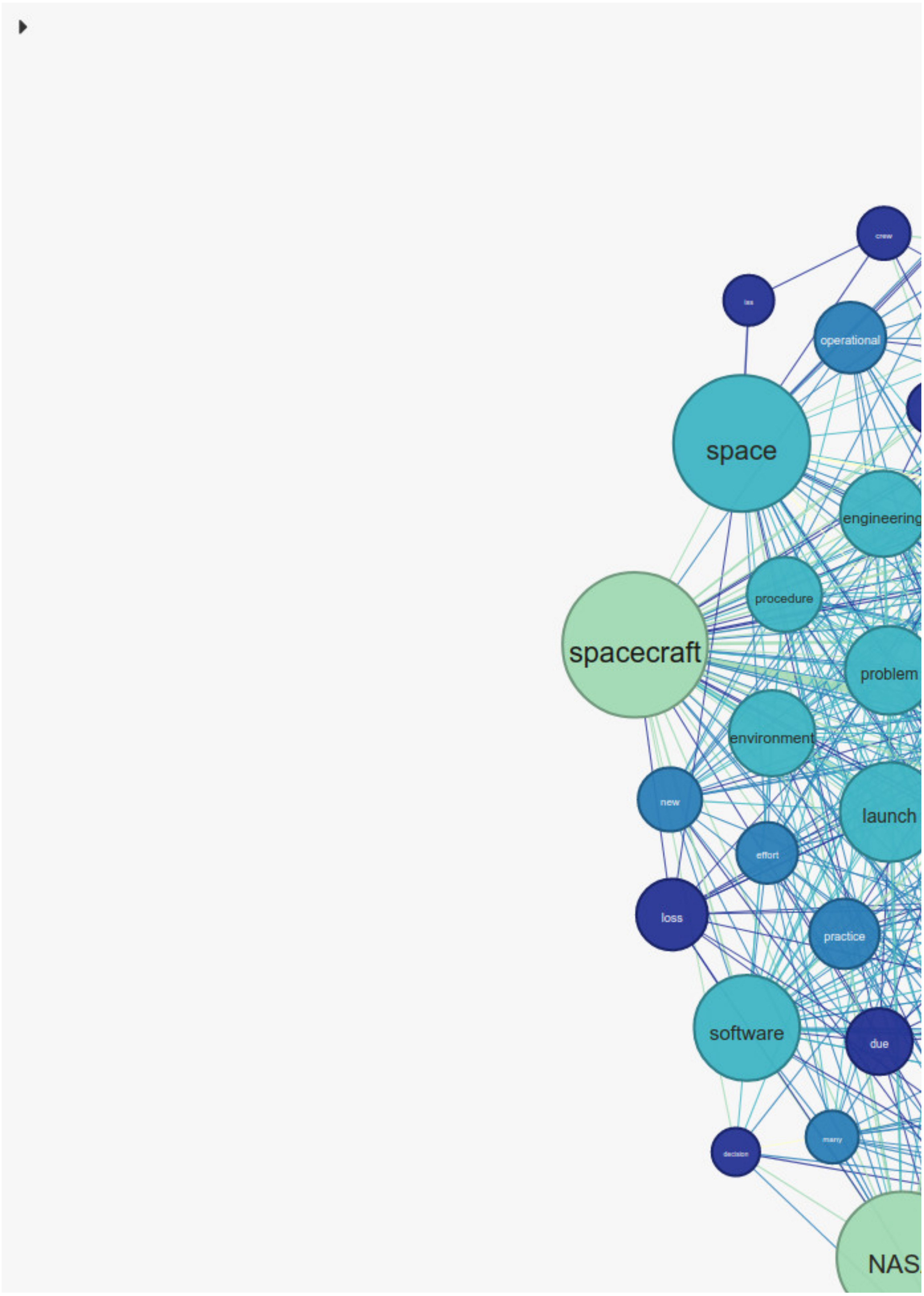
```
“Variable frequency drive systems are installed at the Shuttle launch pads at KSC. The system
allows for a direct coupling between the main propulsion system liquid oxygen pump and drive
motor. This eliminates the motor clutch system, a high maintenance item, and gaseous nitrogen
lines used to purge the clutch system.”
The final keywords and key phrases (meaning the most informative ones, i.e. those that have no
HAS_SUBGROUP relationships within this lesson) identified for this lesson are:
launch pad
propulsion system
drive motor
motor clutch system
frequency drive system
direct coupling
```

Visualization

The keywords extracted can be used in different ways for accessing the entire corpus and discover new insights about the documents. They can be used as an index for the documents in the corpus or even as a way for getting a sort of summary of the content of the documents. The approach that is presented here aims at using the extracted keywords to provide an **advanced visualization** of the corpus allowing people to navigate the content in an easy way. The main idea is to offer a set of tools that, mixing textual searches and visual elements, provide an **assisted search** experience. The visualization follow these rules:

- Every keyword is represented as a bubble
- Relative bubble size is defined by the number of documents the keyword appears in
- A relationship between two bubbles exists if there is a document that is described by both, so they co-occur in that document. Relationship weight is defined by the number of co-occurrences in the corpus.
- A sequential colour gradient + NLP TextRank Gradient- is used to show the degree of the bubble.

This is the result of the previous rules applied to the NASA Lesson Learned :

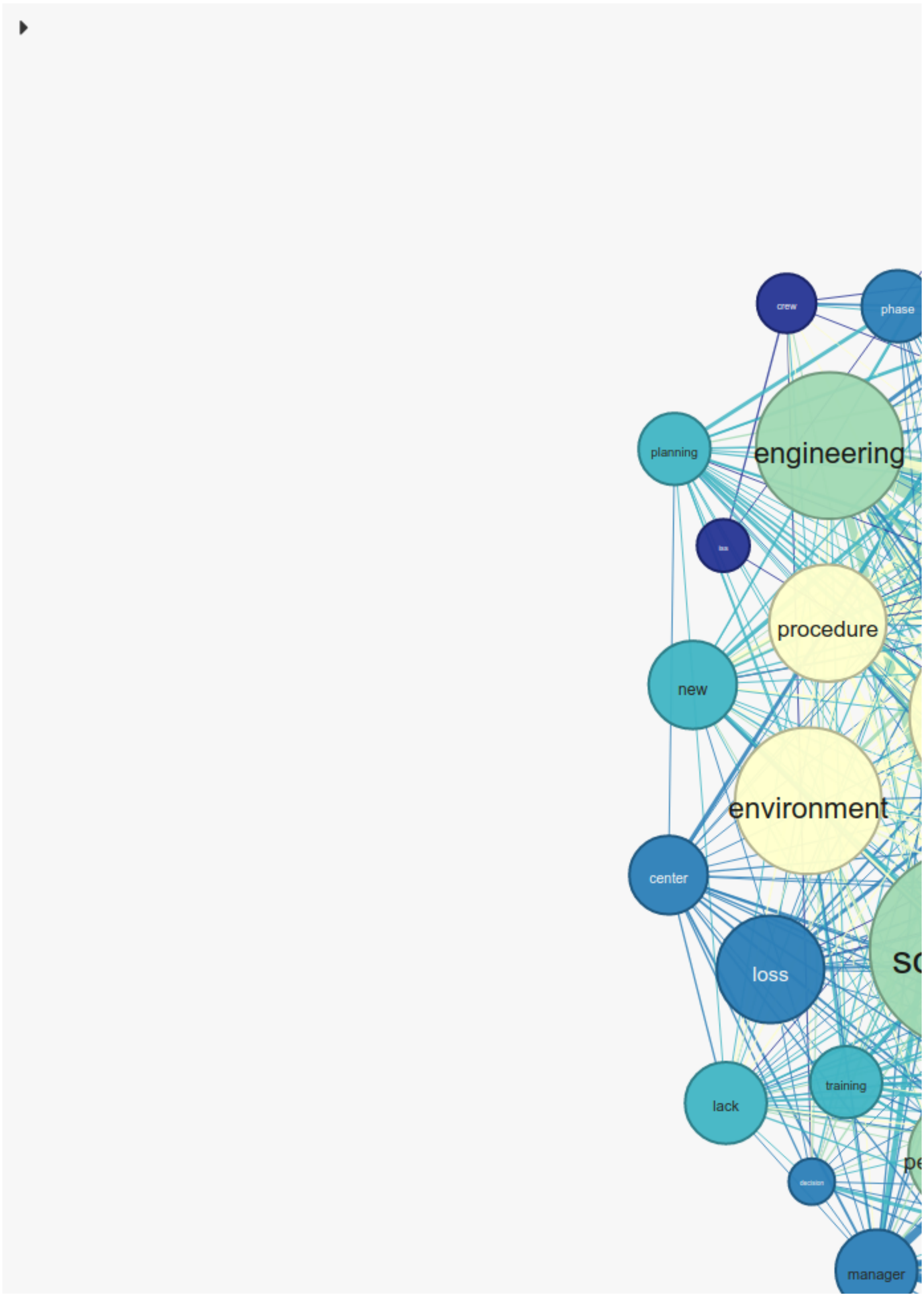

crew
iss
operational
space
engineering
procedure
spacecraft
problem
environment
new
launch
effort
loss
practice
software
due
many
decision
NAS

NLP TextRank Image3

This interface allows users to:

- Select a specific keyword, get the list of related documents, and thoroughly explore their content in a side panel;
- Select multiple keywords to refine the result set considering only the documents related to all the keywords selected;
- Use context actions to quickly add or remove specific bubbles (and their neighbours).

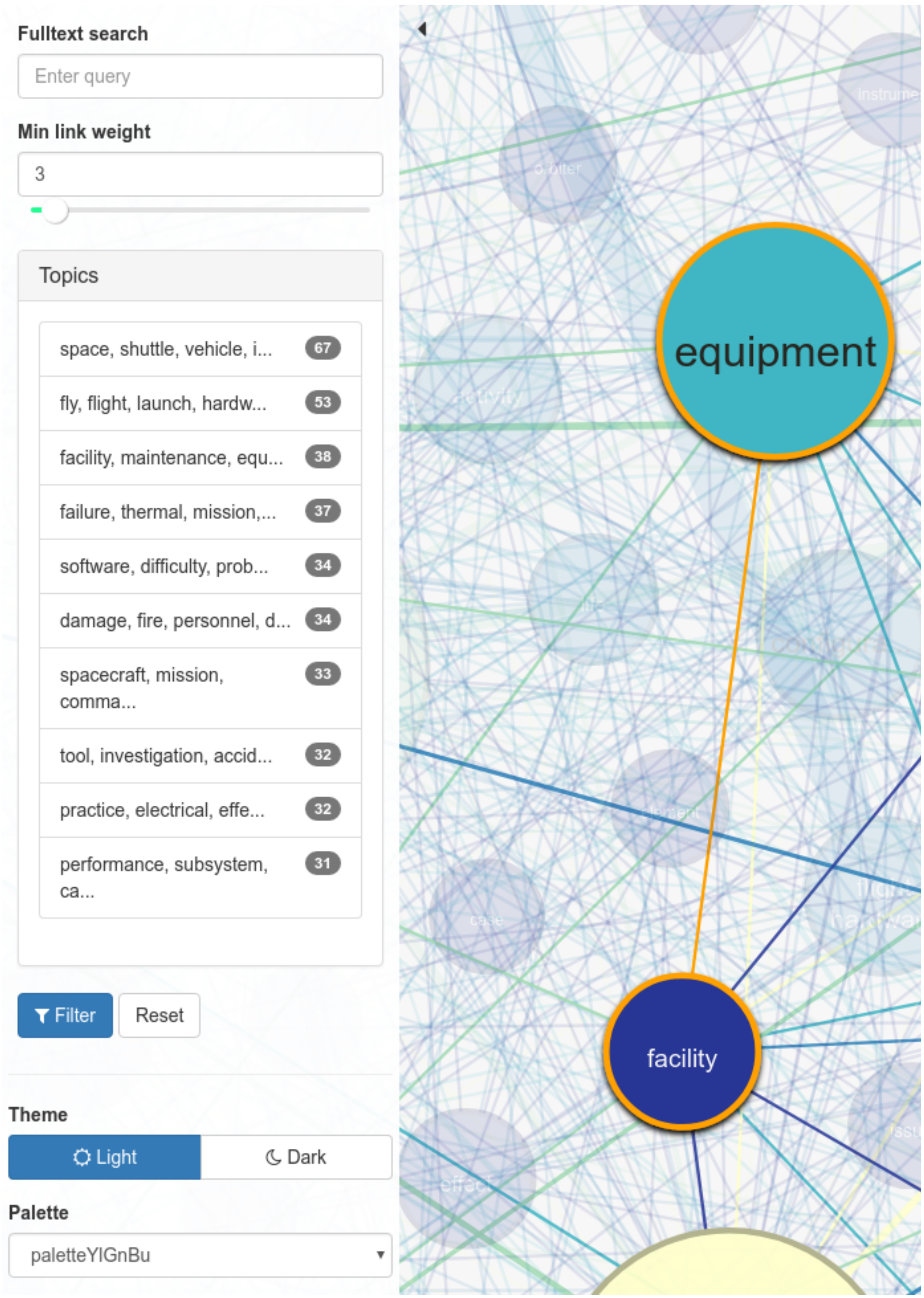
In the following image the biggest bubbles are removed, this means removing the most common keywords and the related connections. This allows users to see more interesting keywords emerging in the graph.


crew
phase
engineering
planning
procedure
new
environment
center
loss
training
lack
decision
manager

NLP TextRank Image4

Some parameters allow refining the navigation of the data:

- A textual search allows to filter documents processed to only those that contain the searched text;
- A threshold allows to define a minimum weight for the co-occurrence relationships;
- A set of layer facets can be applied to refine the results, e.g. topics (automatically extracted using Latent Dirichlet Allocation), data sources, etc.



Fulltext search

Enter query

Min link weight

3

Topics

space, shuttle, vehicle, i...	67
fly, flight, launch, hardw...	53
facility, maintenance, equ...	38
failure, thermal, mission,...	37
software, difficulty, prob...	34
damage, fire, personnel, d...	34
spacecraft, mission, comma...	33
tool, investigation, accid...	32
practice, electrical, effe...	32
performance, subsystem, ca...	31

Filter Reset

Theme

Light Dark

Palette

paletteYlGnBu

NLP TextRank Image5

This is a new, and more effective, approach to document search, extremely useful when the entropy of the documents - in terms of structures, content and even domains - is high. Using this tool, users can access huge and complex corpora even when they don't have a clear idea of what to look for. This visualization is now part of the GraphAware Knowledge Studio.

Conclusion

This blog post describes a specific use case - the keywords extraction of the more complex scenario of analyzing and processing huge set of documents that represent the entire knowledge of an enterprise. Indexing and searching this corpus represents a first step in this direction, but others are required like topic modeling, clustering, domain-specific ontology building, synonyms identification, and so forth. All these allow users to extract insights automatically from the textual data. With them it is possible then to build advanced analytic functionalities for the eventual users, both internal and external, of the organization that owns the set of documents. The Graphaware Knowledge Platform aims at embedding all these functionalities in a harmonised and private environment that hides all the complexities related to ETL, processing, analysis, visualization and management. An intuitive user interface simplifies all these tasks and the API provides easy integration with the current infrastructure. The Graphaware Knowledge Platform is the cognitive computing platform that can be easily "injected" in the enterprises of any size to convert chaos into order.

If you believe GraphAware NLP framework or the Graphaware Knowledge Platform would be useful for your project or organisation, please email us at nlp@graphaware.com and one of our GraphAware team members will get in touch. GraphAware is a Gold sponsor of GraphConnect NYC. Visit our booth for a live demo

Bibliography

https://nlp.stanford.edu/software/dependencies_manual.pdf