Lithium NLP: A System for Rich Information Extraction from Noisy User Generated Text on Social Media

Preeti Bhargava and **Nemanja Spasojevic** and **Guoning Hu**
Lithium Technologies | Klout
San Francisco, CA
preeti.bhargava, nemanja.spasojevic, guoning.hu@lithium.com

Abstract

In this paper, we describe the Lithium Natural Language Processing (NLP) system - a resource-constrained, high-throughput and language-agnostic system for information extraction from noisy user generated text on social media. Lithium NLP extracts a rich set of information including entities, topics, hashtags and sentiment from text. We discuss several real world applications of the system currently incorporated in Lithium products. We also compare our system with existing commercial and academic NLP systems in terms of performance, information extracted and languages supported. We show that Lithium NLP is at par with and in some cases, outperforms state-of-the-art commercial NLP systems.

1 Introduction

Social media has become one of the major means for communication and content production. As a result, industrial systems that possess the capability to process rich user generated content from social media platform have several real-world applications. Furthermore, due to the content style, size and heterogeneity of information (e.g. text, emoticons, hashtags etc.) available on social media, novel NLP techniques and systems that are designed specifically for such content and can potentially integrate or learn information from different sources are highly useful and applicable.

However, NLP on social media data can be significantly complex and challenging due to several reasons:

- **Noisy unnormalized data** - Social media data is much more informal than traditional text and less consistent in language in terms of style, tone etc. It involves heavy usage of slang, jargons, emoticons, or abbreviations which usually do not follow formal grammatical rules. Hence, novel NLP techniques need to be developed for such content.

- **Multi-lingual content** - Social media data poses an additional challenge to NLP practitioners because the user generated content on them is often multi-lingual. Hence, any NLP system processing real world data from the web should be able to support multiple languages in order to be practical and applicable.

- **Large scale datasets** - State-of-the-art NLP systems should be able to work on large scale datasets such as social media data, often involving millions of documents. Moreover, these systems need to have low resource consumption in order to scale to such datasets in a finite amount of time. In addition, in order to be applicable and practical, they should be able to run on off-the-shelf commodity machines.

- **Rich set of information** - In order to be cost-efficient, state-of-the-art NLP systems need to be exhaustive in terms of information extracted[1] from social media text. This includes extracting entities of different types (such as professional titles, sports, activities etc.) in addition to just named entities (such as persons, organizations, locations etc.), inferring

[1] https://en.wikipedia.org/wiki/Information_extraction

Proceedings of the 3rd Workshop on Noisy User-generated Text, pages 131–139

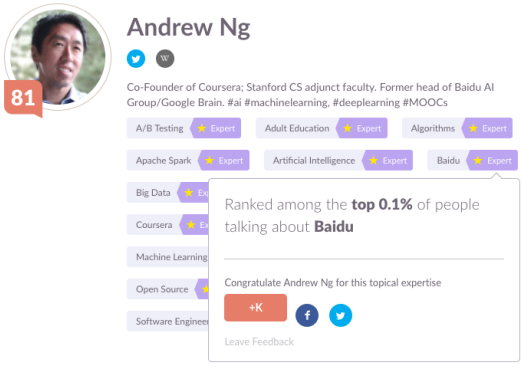

Figure 1: A user's inferred expertise topics

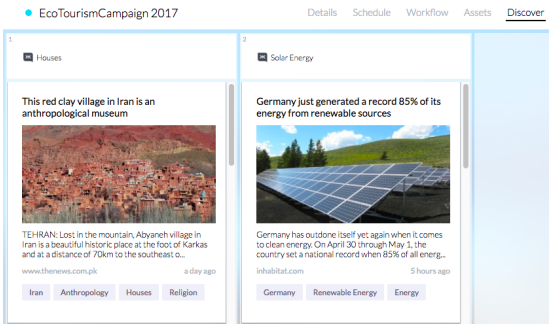

Figure 2: Content Personalization

fine-grained and coarse-grained subject matter topics (sports, politics, healthcare, basketball), text sentiment, hashtags, emoticons etc.

In this paper, we present the Lithium NLP[2] system which addresses these challenges. It is a resource-constrained, high-throughput and language-agnostic system for information extraction from noisy user generated text such as that available on social media. It is capable of extracting a rich set of information including entities, topics, hashtags and sentiment. Lithium NLP currently supports multiple languages including Arabic, English, French, German, Italian and Spanish. It supports large scale data from several social media platforms such as Twitter, Facebook, Linkedin, etc. by processing about $500M$ new social media messages, and $0.5M$ socially relevant URLs shared daily. Since it employs statistical NLP techniques, it uses the large scale of the data to help overcome the noisiness.

Lithium NLP is currently incorporated in several Lithium products. It enables consumer products like Klout[3] - a platform which integrates users' data from multiple social networks such as Twitter, Facebook, Instagram, Linkedin, GooglePlus, Youtube, and Foursquare, in order to measure their online social influence via the *Klout Score*[4] (Rao et al., 2015). On Klout, it is used to model users' topics of interest (Spasojevic et al., 2014) and expertise (Spasojevic et al., 2016) by building their topical profiles. Figure 1 shows an example of a user's topics of expertise, as inferred on Klout. Currently, we build topical profiles for more than $600M$ users. These profiles are further used to recommend personalized content to these users by matching their topics of interest or expertise with content topics as this leads to better user engagement. An example of content personalization is shown in Figure 2. The user scores and topics are also available via the GNIP PowerTrack API[5].

Lithium NLP also enables enterprise products such as Lithium's social media management tools[6] - Lithium Reach and Lithium Response. It is used to analyze $20+M$ new daily engagements across Lithium's 400+ communities[7]. In the past, a version of Lithium NLP had been used to enable user targeting applications such as Klout Perks[8] (influencer reward platform), Cinch[9] (Q&A app), and Who-To-Follow recommendations. These involved selecting a group of users for targeting based on given topics and other filtering criteria.

2 Knowledge Base

Our Knowledge Base (KB) consists of about 1 million Freebase machine ids for entities that were chosen from a subset of all Freebase entities that map to Wikipedia entities. We prefer to use Freebase rather than Wikipedia as our KB since in Freebase, the same id represents a unique entity across multiple languages. Due to limited resources and usefulness of the enti-

[2] A screencast video demonstrating the system is available at https://youtu.be/U-o6Efh6TZc
[3] https://klout.com
[4] https://klout.com/corp/score
[5] http://support.gnip.com/enrichments/klout.html
[6] https://www.lithium.com/products/social-media-management/
[7] https://www.lithium.com/products/online-communities/
[8] https://goo.gl/vtZDqE#Klout_Perks
[9] https://goo.gl/CLcx9p#Cinch

ties, our KB contains approximately 1 million most important entities from among all the Freebase entities. This gives us a good balance between coverage and relevance of entities for processing common social media text. Section 3.1 explains how entity importance is calculated, which enables us to rank the top 1 million Freebase entities.

In addition to the KB entities, we also employ two special entities: **NIL** and **MISC**. **NIL** entity indicates that there is no entity associated with the mention, eg. mention 'the' within the sentence may link to entity **NIL**. This entity is useful especially when it comes to dealing with stop words and false positives. **MISC** indicates that the mention links to an entity which is outside the selected entity set in our KB.

3 System Overview

Figure 3 shows a high level overview of the Lithium NLP system. It has two phases:

3.1 Offline Resource Generation

In this phase, we generate several dictionaries that capture language models, probabilities and relations across entities and topics, by leveraging various multi-lingual data sources. Some of these dictionaries are derived using our DAWT[10] data set (Spasojevic et al., 2017) that consists of densely annotated wikipedia pages across multiple languages. It is 4.8 times denser than Wikipedia and is designed to be exhaustive across several domains.

The dictionaries generated from the DAWT dataset are:

- **Mention-Entity Co-occurrence** - This dictionary captures the prior probability that a mention M_i refers to an entity E_j (including **NIL** and **MISC**) within the DAWT dataset and is equivalent to the cooccurrence probability of the mention and the entity:

$$\frac{count(M_i \rightarrow E_j)}{count(M_i)}$$

  For instance, mention *Michael Jordan* can link to **Michael Jordan (Professor)** or **Michael Jordan (Basketball player)** with different prior probabilities. Moreover, we generate a separate dictionary for each language.

- **Entity-Entity Co-occurrence** - This dictionary captures co-occurrence frequencies among entities by counting all the entities that simultaneously appear within a sliding window of 50 tokens. Moreover, this data is accumulated across all languages and is language independent in order to capture better relations and create a smaller memory footprint when supporting additional languages. Also, for each entity, we consider only the top 30 co-occurring entities which have at least 10 or more co-occurrences across all supported languages. For instance, entity **Michael Jordan (Basketball player)** co-occurs with entities **Basketball**, **NBA** etc. while entity **Michael Jordan (Professor)** co-occurs with entities **Machine Learning**, **Artificial Intelligence**, **UC Berkeley** etc.

We also generate additional dictionaries:

- **Entity Importance** - The entity importance score (Bhattacharyya and Spasojevic, 2017) is derived as a global score identifying how important an extracted entity is for a casual observer. This score is calculated using linear regression with features capturing popularity within Wikipedia links, and importance of the entity within Freebase. We used signals such as Wiki page rank, Wiki and Freebase incoming and outgoing links, and type descriptors within our KB etc.

- **Topic Parents** - This dictionary contains the parent topics for each topic in the Klout Topic Ontology [11] (KTO) - a manually curated ontology built to capture social media users' interests and expertise scores, in different topics, across multiple social networks. As of April 2017, it consists of roughly 8,030 topic nodes and 13,441 edges encoding hierarchical relationships among them.

[10] https://github.com/klout/opendata/tree/master/wiki_annotation

[11] https://github.com/klout/opendata/tree/master/klout_topic_ontology

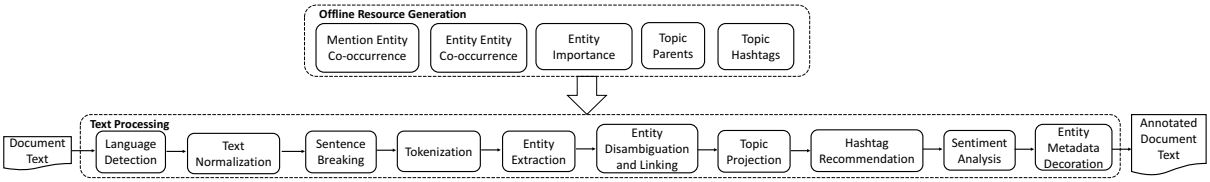

Figure 3: Overview of the Lithium NLP pipeline

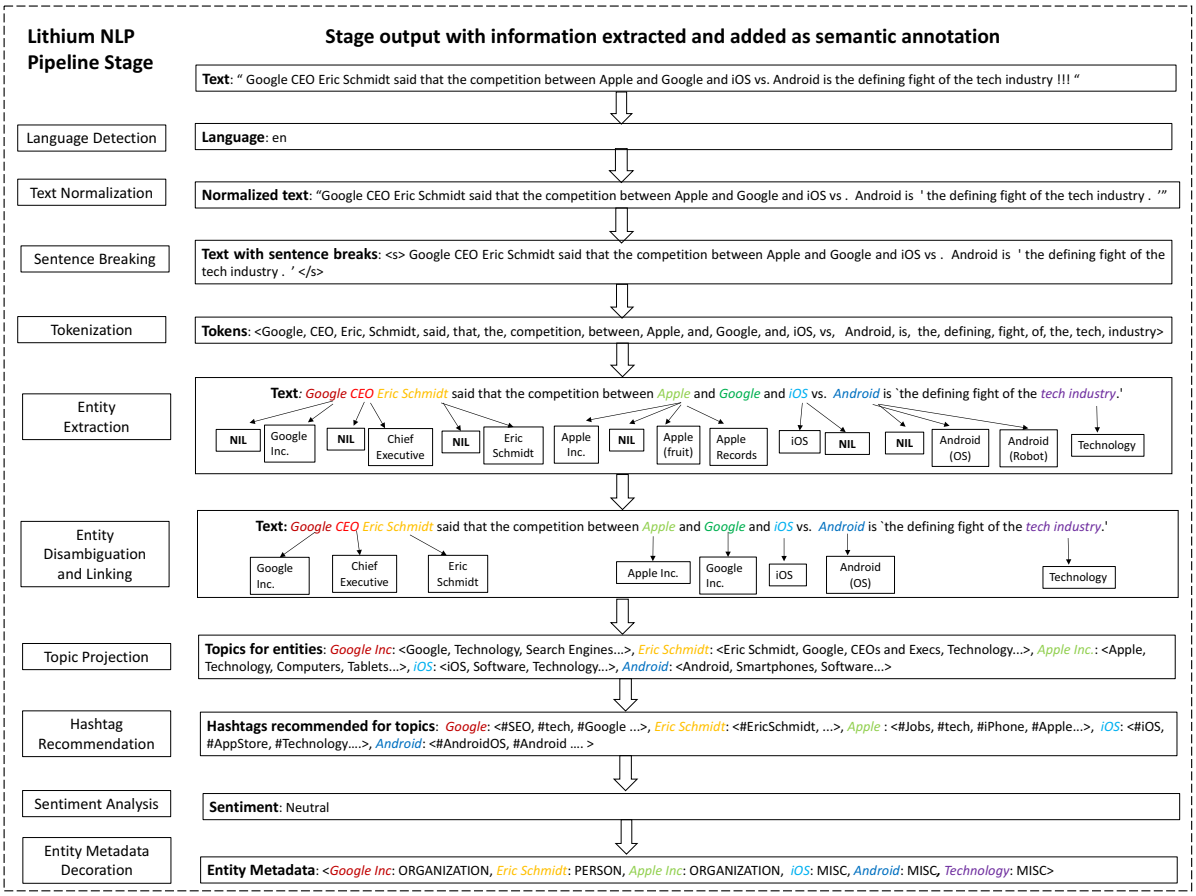

Figure 4: An example demonstrating the information extracted and added as semantic annotation at each stage of the Lithium NLP pipeline (best viewed in color)

- **Topic Hashtags** - This dictionary contains hashtags recommended for topics in KTO. We determine the hashtags via co-occurrence counts of topics and hashtags, importance, recency and popularity of hashtags as well popularity of topics.

3.2 Text Processing

In the Lithium NLP system, an input text document is stored as a Protocol Buffers[12] message. The Text Processing phase of the system processes the input text document through several stages and the information (entities, topics etc.) extracted at every stage is added as a semantic annotation to the text. Not all annotations are added to a document, the Lithium NLP API (explained in Section 3.3) allows a client application to select specific annotations. However, certain annotations such as language and tokens are prerequisites for later stages.

The Text Processing pipeline stages are:

- **Language Detection** - This stage detects the language of the input document using an open source language detector[13]. This detector employs a naive Bayesian

[12] https://developers.google.com/protocol-buffers/

[13] https://github.com/shuyo/language-detection

filter which uses character, spellings and script as features to classify language and estimate its probability. It has a precision of 99% for 49 languages.

- **Text Normalization** - This stage normalizes the text by escaping unescaped characters and replacing special characters (e.g. diacritical marks) based on the detected language. It replaces non-ASCII punctuations and hyphens with spaces, multiple spaces with single space, converts accents to regular characters etc.

- **Sentence Breaking** - This stage breaks the normalized text into sentences using Java Text API[14]. It can distinguish sentence breakers from other marks, such as periods within numbers and abbreviations, according to the detected language.

- **Tokenization** - This stage converts each sentence into a sequence of tokens via the Lucene Standard Tokenizer[15] for all languages and the Lucene Smart Chinese Analyzer[16] for Chinese.

- **Entity Extraction** - This stage extracts mentions in each sentence using the Mention Entity Co-occurrence dictionary generated offline (Section 3.1). A mention may contain a single token or several consecutive tokens, but a token can belong to at most one mention.

  To make this task computationally efficient, we apply a simple greedy strategy that analyzes windows of n-grams (n $\in$ [1,6]) and extracts the longest mention found in each window. For each extracted mention, we generate multiple candidate entities. For instance, mention *Android* can link to candidate entities **Android (OS)** or **Android (Robot)**.

- **Entity Disambiguation and Linking (EDL)** - This stage disambiguates and links an entity mention to the correct candidate entity in our KB (Bhargava et al., 2017). It uses several features obtained from the dictionaries generated offline (Section 3.1). These include context-independent features, such as mention-entity co-occurrence, mention-entity Jaccard similarity and entity importance, and context-dependent features such as entity entity co-occurrence and entity topic semantic similarity. It employs machine learning models, such as decision trees and logistic regression, generated using these features to correctly disambiguate a mention and link to the corresponding entity. This stage has a precision of 63%, recall of 87% and an F-score of 73% when tested on an in-house dataset.

- **Topic Projection** - In this stage, we associate each entity in our KB to upto 10 most relevant topics in KTO. For instance, entity **Android (OS)** will be associated with the topics such as *Smartphones*, *Software* etc.

  We use a weighted ensemble of several semi-supervised models that employ entity co-occurrences, GloVe (Pennington et al., 2014) word vectors, Freebase hierarchical relationships and Wikipedia in order to propagate topic labels. A complete description of this algorithm is beyond the scope of this paper.

- **Hashtag Recommendation** - In this stage, we annotate the text with hashtags recommended based on the topics associated with the text in Topic Projection. This uses the Topic Hashtags dictionary generated offline (Section 3.1)

- **Sentiment Analysis** - In this stage, we determine the sentiment of the text (positive, negative or neutral) via lexicons and term counting with negation handling (Spasojevic and Rao, 2015). For this, we used several lexicons of positive and negative words (including SentiWordNet (Baccianella et al., 2010; Esuli and Sebastiani, 2007) and AFINN (Nielsen, 2011)) as well as emoticons. We compute the sentiment score as

[14] https://docs.oracle.com/javase/7/docs/api/java/text/BreakIterator.html

[15] http://lucene.apache.org/core/4_5_0/analyzers-common/org/apache/lucene/analysis/standard/StandardTokenizer.html

[16] https://lucene.apache.org/core/4_5_0/analyzers-smartcn/org/apache/lucene/analysis/cn/smart/SmartChineseAnalyzer.html

$$\frac{W_{Pos} - W_{Neg}}{\text{Log(Total \# of words in text)} + \epsilon}$$

where W_{Pos} is the weighted strength of positive words and emoticons, W_{Neg} is the weighted strength of negative words and emoticons in the text and ϵ is a smoothing constant. If the score is positive and above a certain threshold, the text is classified as 'Positive'. If it is below a certain threshold, the text is classified as 'Negative'. If it lies within the boundary between 'Positive' and 'Negative' classes, the text is classified as 'Neutral'.

To handle negations, we use a *lookback window*. Every time, we encounter a word from our sentiment lexicons, we look back at a window of size 3 to see if any negation words precede it and negate the weight of the sentiment word. Overall, this stage has a precision of 47%, recall of 48% and an F-score of 46% when tested on an in-house dataset.

- **Entity Metadata Decoration** - In this stage, we add the entity metadata such as its type (Person, Organization, Location, Film, Event, Book) and Location (Population, Time Zone, Latitude/Longitude).

Figure 4 demonstrates how the Lithium NLP pipeline processes a sample text "*Google CEO Eric Schmidt said that the competition between Apple and Google and iOS vs. Android is 'the defining fight of the tech industry'.*" and adds the annotations at every stage.

3.3 REST API

The Lithium NLP system provides a REST API via which client applications can send a text document as request and receive the annotated text as JSON response. A snippet of an annotated response (which is in our text proto format[17]) received through the API is shown in Listing 1. Note that the disambiguated entities are also linked to their Freebase ids and Wikipedia links.

[17] https://github.com/klout/opendata/blob/master/wiki_annotation/Text.proto

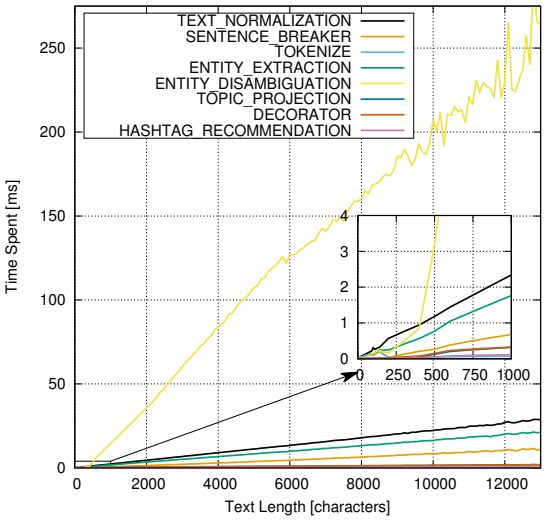

Figure 5: Lithium NLP performance per processing stage (best viewed in color)

Listing 1: JSON of annotated text summary

```
{
  "text": "Vlade Divac Serbian NBA player
      used to play for LA Lakers.",
  "language": "en",
  "annotation_summary": [{
      "type": "ENTITY",
      "annotation_identifier": [{
          "id_str": "01vpr3",
          "id_url": "https://en.wikipedia.org
              /wiki/Vlade_Divac",
          "score": 0.9456,
          "type": "PERSON"
        }, {
          "id_str": "05jvx",
          "id_url": "https://en.wikipedia.org
              /wiki/NBA",
          "score": 0.8496,
          "type": "ORGANIZATION"
        }, ...
      }]
  },
  {
    "type": "KLOUT_TOPIC",
    "annotation_identifier": [{
      "id_str": "6467710261455026125",
      "id_readable": "nba",
      "score": 0.7582
    }, {
      "id_str": "8311852403596174326",
      "id_readable": "los-angeles-lakers",
      "score": 0.66974
    }, {
      "id_str": "8582816108322807207",
      "id_readable": "basketball",
      "score": 0.5445
    }, ...]
  },
  {
    "type": "HASHTAG",
    "annotation_identifier": [{
      "id_str": "NBA",
      "score": 54285.7515
    }, {
      "id_str": "NBAPlayoffs",
      "score": 28685.6006
    }, ...]
  }],
   "sentiment": 0.0
}
```

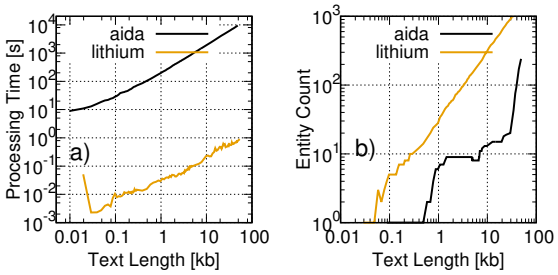

Figure 6: AIDA vs. Lithium NLP Comparison on **a)** Text processing runtime **b)** Extracted entity count (best viewed in color)

3.4 Performance

Figure 5 shows the computational performance per processing stage of the Lithium NLP system. The overall processing speed is about 22ms per 1kb of text. As shown, the time taken by the system is a linear function of text size. The EDL stage takes about 80% of the processing time.

4 Comparison with existing NLP systems

Currently, due to limited resources at our end and also due to inherent differences in the Knowledge Base (Freebase vs Wikipedia and others), test dataset, and types of information extracted (entities, topics, hashtags etc.), a direct comparison of the Lithium NLP system's performance (in terms of precision, recall and f-score) with existing academic and commercial systems such as Google Cloud NL API[18], Open Calais[19], Alchemy API[20], Stanford CoreNLP[21] (Manning et al., 2014), Ambiverse/AIDA[22] (Nguyen et al., 2014) and Twitter NLP[23] (Ritter et al., 2011, 2012) is not possible. Hence, we compare our system with some of them on a different set of metrics.

4.1 Comparison on runtime and entity density

We compare the runtime of Lithium NLP and AIDA across various text sizes. As shown in Figure 6, Lithium NLP is on an average 40,000 times faster than AIDA whose slow runtime can be attributed mainly to Stanford NER. In addition to speed, we also compare the number of entities extracted per kb of text. As shown, Lithium NLP extracts about 2.8 times more entities than AIDA.

4.2 Comparison on information extracted

Table 1 compares the types of information extracted by Lithium NLP system with existing systems. In this comparison, we explicitly differentiate between named entities (Person, Location etc.) and other entity types (Sports, Activities) as well as fine-grained topics (Basketball) and coarse-grained topics (Sports) to demonstrate the rich set of information extracted by Lithium NLP. As evident, most other systems do not provide the rich set of semantic annotations that Lithium NLP provides. A majority of the systems focus on recognizing named entities and types with only a few focusing on sentiment and coarse-grained topics as well. In contrast, Lithium NLP extracts, disambiguates and links named and other entities, extracts subject matter topics, recommends hashtags and also infers the sentiment of the text.

4.3 Comparison on languages

Table 2 compares the languages supported by the Lithium NLP system with existing systems. As evident, Lithium supports 6 different languages which is at par and in some cases, more than existing systems.

5 Conclusion and Future Work

In this paper, we described the Lithium NLP system - a resource-constrained, high-throughput and language-agnostic system for information extraction from noisy user generated text on social media. Lithium NLP extracts a rich set of information including entities, topics, hashtags and sentiment from text. We discussed several real world applications of the system currently incorporated in Lithium products. We also compared our system with existing commercial and academic NLP systems in terms of performance, information extracted and languages supported. We showed that Lithium NLP is at par with and in some

[18] https://cloud.google.com/natural-language/
[19] http://www.opencalais.com/opencalais-demo/
[20] https://alchemy-language-demo.mybluemix.net/
[21] http://corenlp.run/
[22] https://www.ambiverse.com/
[23] https://github.com/aritter/twitter_nlp

	Lithium NLP	Google NL	Open Calais	Alchemy API	Stanford CoreNLP	Ambiverse	Twitter NLP
Named Entities	X	X	X	X	X	X	X
Other Entities	X	X	X	X	X		
Topics (fine-grained)	X						
Topics (coarse-grained)	X		X	X			
Hashtags	X						
Document Sentiment	X			X	X		
Entity level Sentiment		X		X			
Entity types	X	X	X	X	X	X	X
Relationships			X	X	X		
Events							X

Table 1: Comparison of information extracted by Lithium NLP with existing NLP systems

	Lithium NLP	Google NL	Open Calais	Alchemy API	Stanford CoreNLP	Ambiverse	Twitter NLP
Supported Languages	Arabic, English, French, German, Italian, Spanish	Chinese, English, French, German, Italian, Japanese, Korean, Portugese, Spanish	English, French, Spanish	English, French, German, Italian, Portuguese, Russian, Spanish, Swedish	Arabic, Chinese, English, French, German, Spanish	English, German, Spanish, Chinese	English

Table 2: Comparison of languages supported by Lithium NLP with existing NLP systems

cases, outperforms state-of-the-art commercial NLP systems.

In future, we plan to extend the capabilities of Lithium NLP to include entity level sentiment as well. We also hope to collaborate actively with academia and open up the Lithium NLP API to academic institutions.

Acknowledgements

The authors would like to thank Prantik Bhattacharya, Adithya Rao and Sarah Ellinger for their contributions to the Lithium NLP system. They would also like to thank Mike Ottinger and Armin Broubakarian for their help with building the Lithium NLP UI and demo.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Preeti Bhargava, Nemanja Spasojevic, and Guoning Hu. 2017. High-throughput and language-agnostic entity disambiguation and linking on user generated data. In *Proceedings of WWW 2017 workshop on Linked Data on the Web*.

Prantik Bhattacharyya and Nemanja Spasojevic. 2017. Global entity ranking across multiple languages. In *Companion Proceedings of the WWW*, pages 761 – 762.

Andrea Esuli and Fabrizio Sebastiani. 2007. Sentiwordnet: A high-coverage lexical resource for opinion mining. *Evaluation*, pages 1–26.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014. Aida-light: High-throughput named-entity disambiguation. In *LDOW'14*.

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532 – 1543.

Adithya Rao, Nemanja Spasojevic, Zhisheng Li, and Trevor Dsouza. 2015. Klout score: Measuring influence across multiple social networks. In *IEEE Intl. Conf. on Big Data*.

Alan Ritter, Mausam Clark, Sam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Empirical Methods in Natural Language Processing*.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *KDD*.

Nemanja Spasojevic, Preeti Bhargava, and Guoning Hu. 2017. Dawt: Densely annotated wikipedia texts across multiple languages. In *Companion Proceedings of WWW*, pages 1655 – 1662.

Nemanja Spasojevic, Prantik Bhattacharyya, and Adithya Rao. 2016. Mining half a billion topical experts across multiple social networks. *Social Network Analysis and Mining*, 6(1):1–14.

Nemanja Spasojevic and Adithya Rao. 2015. Identifying actionable messages on social media. In *IEEE International Conference on Big Data*, IEEE BigData '15.

Nemanja Spasojevic, Jinyun Yan, Adithya Rao, and Prantik Bhattacharyya. 2014. Lasta: Large scale topic assignment on multiple social networks. In *Proc. of ACM KDD*, pages 1809 – 1818.