GloGen: Automated GLOssary GENeration for effective and efficient information extraction from text data (Challenge 20)

Jack Ragless
jack.ragless@student.adelaide.edu.au
School of Computer Science
Adelaide, SA, Australia

Dr Christoph Treude
christoph.treude@adelaide.edu.au
School of Computer Science
Adelaide, SA, Australia

1 INTRODUCTION

Knowledge contained within textual data can hold key information to enable superior decision making. However, extracting the right knowledge and deriving evidence required by analysts and decision makers is often dependent on creating domain-specific glossaries that are both time consuming to develop and difficult to port across multiple applications. Applying Artificial Intelligence techniques (within Natural Language Processing and the broader area) can inject efficiency to this process by automating the building process for glossaries. For the purpose of this problem, a glossary is defined as a content analysis dictionary that can be referenced to extract relevant information from a corpus. It should ideally have a structured list of terms, phrases and concepts relevant to the terms arranged in some specified category.

We have addressed this challenge by developing GloGen, a system which first identifies which terms should be defined in a corpus, and then provides domain-specific definitions for the terms in the glossary. Importantly, the same term will be defined differently depending on the domain of the input corpus. For example, GloGen successfully can distinguish the keyword 'evolve' in the context of an article about Charles Darwin (change genetic composition) and in the context of an article about children (develop).

The next sections detail the implementation and preliminary evaluation of GloGen. The source code and documentation are available at https://github.com/jackragless/GLOGEN-automatic-GLOssary-GENerator.

2 KEYWORD EXTRACTION AND ABSTRACTIVE SUMMARISATION

Glossary generation consists of two primary problems requiring prioritisation:

- Which terms to define?
- How to define a given term?

Most existing keyword extraction techniques that could be employed to address the first problem (Which terms to define?) use rule-based (primarily based on syntax), statistical (e.g., Term Frequency - Inverse Document Frequency (TF-IDF), RAKE/YAKE) or graph-based methods (TextRank [15, 16]). In SemEval 2010, all traditional models achieved an F_1-score ≥ 0.275 [14]. They performed similarly in an array of other benchmark corpora, with none achieving a result greater than 0.5 [12]. This led us to consider deep-learning keyword extraction methods as an alternative. Note the academic literature in this area is sparse. A 2017 paper by Zhang et al. shows an improved SemEval 2010 score of 0.308 using a custom Convolutional Neural Network (CopyCNN) [19]. This has since been outperformed by Google's Bidirectional Encoder Representations from Transformers (BERT) [13], achieving an F_1-score of 0.57 [3]. Based on these findings we adopted a transformers-based, deep-learning approach for addressing the first problem.

For the second problem (How to define a given term?), we initially considered using an abstractive summarisation technique for generating definitions from source text. Our review into this field, however, indicated it was unlikely to produce accurate definitions from documents potentially containing thousands of words [17]. For example, Zhang et al.'s state-of-the-art PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive Summarisation) aims only to abstract masked single words within individual sentences [18]. Hence, we decided to source definition sentences from existing online sources.

3 GLOGEN DESIGN AND IMPLEMENTATION

This section is a conceptual explanation of GloGen's structure, visualised in Figure 1. For technical details on how to run GloGen refer to the repository's README file.

GloGen is split into two distinct modules:

- Wikipedia corpus miner / machine learning keyword extraction training (Module 1)
- Glossary generator (Module 2)

3.1 Module 1

The basic intuition behind Module 1 is to use wikilinks inside of Wikipedia articles as training data for which keywords from a corpus should be included in a glossary. Presumably, keywords and keyphrases which link to other Wikipedia pages are worth defining. We detail this process in the following paragraphs, along with several filter steps we employed to improve the output.

3.1.1 Wikipedia Corpus Miner. The Wikipedia Corpus Mineer generates a corpus of Wikipedia page objects (structure: 'title, text, keywords') for a given domain. It achieves this utilising Wikipedia's Category system [10]. Each category page contains additional subcategories and/or regular pages. All regular pages are added to the corpus, then the subcategories are parsed. This is performed recursively until only regular pages remain in a category or a set 'depth' limit is reached. The depth setting affects corpus size and specificity, as deeper pages tend to contain less-generalised content. See Figure 2 for an example of linked terms in Wikipedia.

In each page hyperlinks to other Wikipedia pages are regarded as keywords/phrases.

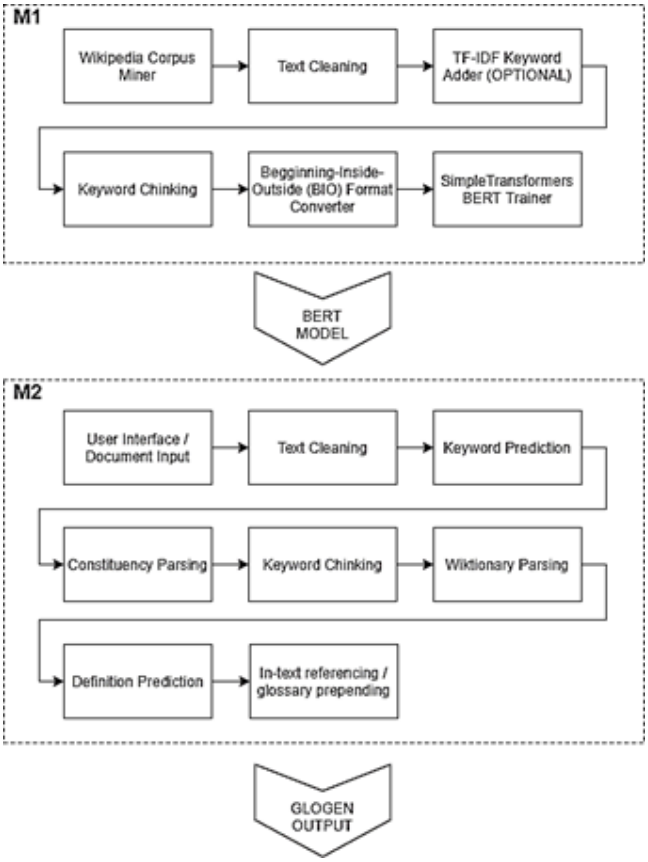

Figure 1: GloGen module 1 and 2 pipeline

Despite Wikipedia's shortcomings, we deemed it suitable for building a keyword extraction corpus thanks to its large size (approximately 6.2M English pages at the time of writing [11]), aforementioned category structure and keyword highlighting. Digital book corpora such as Project Gutenberg were disregarded as it is difficult to automatically detect books with keyword identification and then determine the formatting mechanism (e.g., highlighting, boldening, italics etc).

3.1.2 Text cleaning. Currently the cleaning function is configured to remove various formatting symbols, consecutive whitespacing and foreign symbols from a given Wikipedia text. The cleaning function also has the ability to remove stopwords, remove all grammatical symbols, lemmatise and lowercase, but these are disabled at present. We discovered that leaving these elements intact allows the model to better determine valid keyword phrases. For instance, when grammatical symbols were originally removed, the AI model tended to combine separate items into a single phrase which were previously comma-separated.

3.1.3 (OPTIONAL) TF-IDF Keyword Adder. This stage manually adds keywords to a Wikipedia object based on TF-IDF ranking. This is intended to overcome the shortcomings of Wikipedia authors who occasionally fail to hyperlink esoteric terms. It also adds any non-hyperlinked acronyms.

```
Computer science is the study of
algorithmic processes, computational machines and
computation itself.
```

Figure 2: Example of Wikipedia's hyperlinks

```
[[1, 'Computer', 'O'],  [1, 'science', 'O'],
[1, 'is', 'O'],         [1, 'the', 'O'],
[1, 'study', 'O'],      [1, 'of', 'O'],
[1, 'algorithmic', 'B'],[1, 'processes', 'I'],
[1, ',', 'O'],          [1, 'computational', 'B'],
[1, 'machines', 'I'],   [1, 'and', 'O'],
[1, 'computation', 'O'],[1, 'itself', 'O'],
[1, '.', 'O']]
```

Figure 3: Example of BIO formatted sentence

3.1.4 Keyword Chinking. Keyword Chinking refers to a heuristic system to remove classes of keywords/phrases we deemed unnecessary to define. Currently, these classes include:

- Companies and educational institutions [1]
- People [4]
- Geographic locations (using Spacy NER)
- Common words (top 10k from Google's Trillion Word Corpus [2])

Depending on context and user preference, these can be individually disabled within the chinking script.

3.1.5 BIO Format Converter. The BIO Format Converter tokenises all words from all Wikipedia page objects into a singular array and labels them as 'Beginning', 'Inside' or 'Outside' depending on whether it is part of a keyword/keyphrase. This [sentence_number, word, label] data format is required by the SimpleTransformers trainer [5]. See Figure 3 for an example.

3.1.6 SimpleTransformers Trainer. The default starting model is 'bert-base-cased'. We selected this model as it is the original version of BERT; not biased towards any individual application. We then tweaked our training data to achieve the best possible F_1-score with this baseline (see Results for metrics). Future users, however, may wish to improve performance by implementing other models such as DistilBERT for faster training or BERT-Large for additional encoder layers.

The output of the machine learning keyword extractor is used in Module 2.

3.2 Module 2

3.2.1 User Interface. The user interface allows users to direct to the directory where .txt files are stored. This component reads in title and text from the .txt files and performs the steps below.

3.2.2 Text Cleaning. The text cleaning step applies the same cleaning function applied to the training corpus for consistency (see Module 1; Step 2).

```
{noun:{
{def: '(general) An object designed to open
    and close a lock.',
ex: 'She used the key to unlock the house.'},
{def: '(general) A crucial step or requirement.',
ex: 'The key to solving this problem is
    persistence.'}
},
adjective:{
        {def: '(general) Important, salient.',
        ex: 'She makes several key points.'}
}}
```

Figure 4: Example of Wiktionary Parser output

3.2.3 Keyword Prediction. The BERT model from Module 1 is used to predict candidate keywords/phrases within the text.

3.2.4 Constituency Parsing. Verb and noun phrases and individual part-of-speech tags are parsed from constituency trees.

3.2.5 Keyword Chinking. Occasionally names, locations, common words, etc. will still be output by the keyword extraction model despite their omission from original training dataset. These filters are reapplied here to remove these terms from the predicted list. Additionally, any predicted keyphrases (containing two or more words) which do not appear as verb phrases or noun phrases during constituency parsing are removed.

3.2.6 Wiktionary Parsing. Wiktionary Parsing searches for a predicted term in Wiktionary (https://www.wiktionary.org/) and parses all resulting definitions into part-of-speech categories. Each part-of-speech contains definition-example pairs. Prepended to each definition string is a bracket containing the definition domain if provided. At default, domain equals 'general'. This code is a modified version of the WiktionaryParser library [8]. See Figure 4 for an example of the output of the Wiktionary Parser.

Wiktionary was chosen as the external definition source due to its:

- High definition number (approximately 1.3M at time of writing [7])
- Wide topic coverage
- Part-of-speech structure
- Existing Python packages (WiktionaryParser)
- Creative Commons Attribution-ShareAlike 3.0 Unported License

Copyright was the most important consideration, with competitors such as Dictionary.com and The Free Dictionary strictly prohibiting mining and redistribution of their data.

3.2.7 Definition Prediction. First, the constituency parse data is used to select definitions from the correct part-of-speech category. Next, the top two definitions are determined based on the primary prediction mechanism; the semantic similarity score between the definition sentence and the original document (using semantic-text-similarity [6]). A secondary prediction mechanism, either semantic similarity of example / source text or embedding similarity of domain / source text title, is chosen based on data availability. In other words, if more examples are available, choose the former; if more domains are available, select the latter. The definition which scores top two in both prediction metrics is returned. If this condition is unmet the prediction defaults to the primary predictor's top selection. This mechanism is designed to avoid penalising definitions which lack example / domain data. The data justifying this composite prediction mechanism is provided in the Results section of this report.

3.2.8 In-text Referencing / Glossary Prepending. Numbered references are inserted into the original text which correspond to the prepended glossary. The final output is saved locally to .txt files in directory/GLOGEN.

4 EXAMPLE OF GLOGEN GLOSSARY OUTPUT

This section shows the output of GloGen for the Bitcoin Wikipedia article at https://en.wikipedia.org/wiki/Bitcoin.

4.1 Generated Glossary

|1| Bitcoin: Alternative letter-case form of bitcoin. A unit of the bitcoin digital currency.

|2| cryptocurrency: A digital currency based on a cryptographic system.

|3| distributed ledger: A distributed ledger (also called a shared ledger or distributed ledger technology or DLT) is a consensus of replicated, shared, and synchronized digital data geographically spread across multiple sites, countries, or institutions.

|4| Bitcoins: plural of Bitcoin. Alternative letter-case form of bitcoin.

|5| University Cambridge: The University of Cambridge (legally The Chancellor, Masters, and Scholars of the University of Cambridge, also known as Cambridge University) is a collegiate research university in Cambridge, United Kingdom.

|6| speculative bubble: An economic bubble or asset bubble (sometimes also referred to as a speculative bubble, a market bubble, a price bubble, a financial bubble, a speculative mania, or a balloon) is a situation in which asset prices appear to be based on implausible or inconsistent views about the future.

|7| mined: simple past tense and past participle of mine. To earn new units of cryptocurrency by doing certain calculations.

|8| coinbase: Coinbase is a digital currency exchange headquartered in San Francisco, California, United States.

|9| timestamp: A variable containing the date and time at which an event occurred, often included in a log to track the sequence of events.

|10| instability: The quality of being unstable.

|11| cypherpunk: A person with an interest in encryption and privacy, especially one who uses encrypted email.

|12| crashed: simple past tense and past participle of crash. To cause to terminate extraordinarily.

|13| LevelDB: LevelDB is an open-source on-disk key-value store written by Google fellows Jeffrey Dean and Sanjay Ghemawat.

|14| blockchain: A shared record of past transactions in a cryptocurrency network.

|15| OpenSSL: OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end.

|16| blockchains: plural of blockchain. A shared record of past transactions in a cryptocurrency network.

4.2 Modified Input Document

The following shows the modified input document for an excerpt of the Wikipedia article of Bitcoin. Note the numbers inside || which refer to the corresponding glossary entry.

Bitcoin|1| (₿) is a cryptocurrency|2| invented in 2008 by an unknown person or group of people using the name Satoshi Nakamoto. The currency began use in 2009 when its implementation was released as open-source software.Bitcoin is a decentralized digital currency, without a central bank or single administrator, that can be sent from user to user on the peer-to-peer bitcoin network without the need for intermediaries. Transactions are verified by network nodes through cryptography and recorded in a public distributed ledger|3| called a blockchain. Bitcoins|4| are created as a reward for a process known as mining. They can be exchanged for other currencies, products, and services. Research produced by the University of Cambridge estimated that in 2017, there were 2.9 to 5.8 million unique users using a cryptocurrency wallet, most of them using bitcoin.Bitcoin has been criticized for its use in illegal transactions, the large amount of electricity used by miners, price volatility, and thefts from exchanges. Some economists, including several Nobel laureates, have characterized it as a speculative bubble|6| at various times. Bitcoin has also been used as an investment, although several regulatory agencies have issued investor alerts about bitcoin.

== History ==

=== Creation ===

The domain name "bitcoin.org" was registered on 18 August 2008. On 31 October 2008, a link to a paper authored by Satoshi Nakamoto titled Bitcoin: A Peer-to-Peer Electronic Cash System was posted to a cryptography mailing list. Nakamoto implemented the bitcoin software as open-source code and released it in January 2009. Nakamoto's identity remains unknown.On 3 January 2009, the bitcoin network was created when Nakamoto mined|7| the starting block of the chain, known as the genesis block. Embedded in the coinbase|8| of this block was the text "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks". This note references a headline published by The Times and has been interpreted as both a timestamp|9| and a comment on the instability|10| caused by fractional-reserve banking.The receiver of the first bitcoin transaction was cypherpunk|11| Hal Finney, who had created the first reusable proof-of-work system (RPoW) in 2004. Finney downloaded the bitcoin software on its release date, and on 12 January 2009 received ten bitcoins from Nakamoto. Other early cypherpunk supporters were creators of bitcoin predecessors: Wei Dai, creator of b-money, and Nick Szabo, creator of bit gold.

=== 2011–2012 ===

By 9 January the price had risen to $7.38, but then crashed|12| by 49% to $3.80 over the next 16 days. The price then rose to $16.41 on 17 August, but fell by 57% to $7.10 over the next three days.The Bitcoin Foundation was founded in September 2012 to promote bitcoin's development and uptake.On 1 November 2011, the reference implementation Bitcoin-Qt version 0.5.0 was released. It introduced a front end that used the Qt user interface toolkit. The software previously used Berkeley DB for database management. Developers switched to LevelDB|13| in release 0.8 in order to reduce blockchain|14| synchronization time. The update to this release resulted in a minor blockchain fork on the 11 March 2013. The fork was resolved shortly afterwards. Seeding nodes through IRC was discontinued in version 0.8.2. From version 0.9.0 the software was renamed to Bitcoin Core. Transaction fees were reduced again by a factor of ten as a means to encourage microtransactions. Although Bitcoin Core does not use OpenSSL|15| for the operation of the network, the software did use OpenSSL for remote procedure calls. Version 0.9.1 was released to remove the network's vulnerability to the Heartbleed bug.

Table 1: BERT training metrics

epoch	evaluation loss	precision	recall	F_1-score
1	0.116103	0.771397	0.832356	0.800718
2	0.100975	0.799877	0.857138	0.827518
3	0.094517	0.827228	0.858880	0.842757
4	0.096531	0.820281	0.873003	0.845821
5	0.097755	0.838606	0.865873	0.852022
6	0.103284	0.829084	0.872611	0.850291
7	0.111309	0.843595	0.873736	0.858401
8	0.118182	0.839704	0.875811	0.857377
9	0.119943	0.847391	0.873944	0.860463
10	0.124928	0.850195	0.872474	0.861190
11	0.142617	0.845942	0.873042	0.859278
12	0.162345	0.853155	0.874262	0.863580
13	0.162984	0.850092	0.876520	0.863103
14	0.177804	0.848393	0.878457	0.863163
15	0.188152	0.854893	0.876523	0.865573
16	0.199405	0.850014	0.882772	0.866084
17	0.200429	0.855467	0.878026	0.866600
18	0.216157	0.854457	0.880877	0.867466

5 EXPERIMENTAL SETUP AND RESULTS

Module 1 was trained on Wikipedia's 'Category: Computer Science'. This domain yielded approximately 42k pages at a mining depth of three.

We cleaned page text for formatting symbols, consecutive whitespacing and foreign symbols. Removal of grammatical symbols, stopword removal, lemmatisation and lowercasing were disabled to provide contextual clues to the model during training. Keywords were derived only from Wikipedia hyperlinks (i.e., the optional TF-IDF adder was disabled during this test).

Lastly, we used the default 'bert-base-cased' model in SimpleTransformers, with epochs set to 18 and a batch size of 32. As seen in Figure 5 and Table 1, the model achieved a best F_1-score of 0.867. Note, however, that iterative improvement plateaus after epoch 3. This pretrained Computer Science domain model is provided with GloGen.

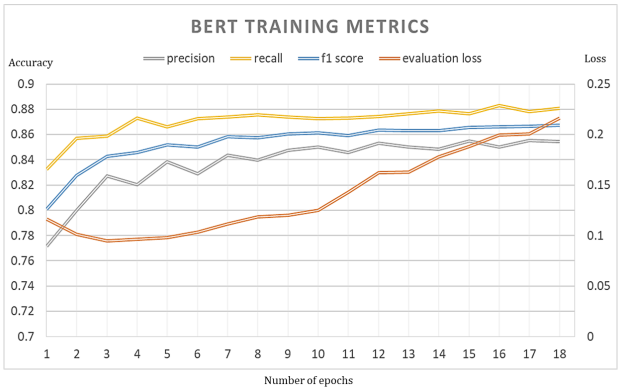

Figure 5: BERT training metrics

Table 2: Wiktionary prediction accuracy

prediction method	accuracy (%)
domain↔title (*embeddings)	38
example↔text	42
definition↔text	76
definition↔text + example↔text	78
definition↔text + domain↔text	78
definition↔text + composite	80

Measuring the overall quality of GloGen's glossary output is only possible via a blind survey. In lieu of this subjective process, we created a test dataset for measuring the objective accuracy of various definition prediction mechanisms. This dataset consists of 25 domain variant term pairs (50 total test cases) and is largely based on a publicly available dataset of terms that have different meanings for scientists and the public [9]. Each has a keyword, Wikipedia page (source text), part-of-speech and actual definition. If the predicted and actual definition match this contributes to a model's final accuracy score. As an example, in Figure 6 the best performing prediction models could correctly differentiate between the biological and generalised form of the term 'evolve'. The figure contains three more examples of such definition pairs, and the entire dataset is available in GloGen's repository.

The 'composite' mechanism described in Module 2; Step 7 achieved 80% accuracy, the highest of all mechanisms tested. The core of its performance lies in the definition↔text semantic prediction mechanism, but it manages an additional 4% performance by considering a definition's domain or usage example. This is the prediction mechanism included in GloGen.

In Table 2, L↔R, the L attribute is taken from Wiktionary data, while the R attribute is taken from the original document. For example, 'example↔text' is the semantic similarity score between a candidate definition's example and the source document.

Note this 80% accuracy level may be lower in GloGen's final glossary input due to:

- Berkeley's Neural Parser's (Benepar) part-of-speech predictions for words/phrases

```
{wiki: Charles Darwin, keyword: evolve, pos:
    verb,
 DEF1: (biology) Of a population, to change
    genetic composition over successive
    generations through the process of
    evolution.},
{wiki: Children, keyword: evolve, pos: verb,
 DEF1: (intransitive) To change; transform.
 DEF2: To come into being; develop.},

{wiki: Pest Control, keyword: aerosol,
    pos: noun,
 DEF1: The payload (eg insecticide, paint,
    oil, cosmetics) and propellant contained
    by an aerosol can.},
{wiki: Physics, keyword: aerosol, pos: noun,
 DEF1: A mixture of fine solid particles or
    liquid droplets suspended in a gaseous
    medium.
 DEF2: (physics) A colloidal system in which
    the dispersed phase is composed of either
    solid or liquid particles and in which
    the dispersal medium is some gas, usually
    air.},

{wiki: Chemistry, keyword: assay, pos: noun,
 DEF1: Examination and determination; test.
 DEF2: The qualitative or quantitative
    chemical analysis of something.},
{wiki: Law, keyword: assay, pos: noun,
 DEF1: Trial, attempt.
 DEF2: Trial by danger or by affliction;
    adventure; risk; hardship; state of
    being tried.},

{wiki: Psychology, keyword: bias, pos: noun,
 DEF1: (countable, uncountable) Inclination
    towards something.},
{wiki: Statistics, keyword: bias, pos: noun,
 DEF1: (statistics) The difference between
    the expectation of the sample estimator
    and the true population value, which
    reduces the representativeness of the
    estimator by systematically distorting
    it.}
```

Figure 6: Domain variant term pairs

- Wiktionary's coverage and community. If a usage of a term is missing, semantic prediction will select from whatever is available. If the term is unavailable in Wiktionary it will not show in the glossary.

6 LIMITATIONS AND FUTURE WORK

GloGen's primary limitation is that outcomes 2 and 3 of DAIC's challenge specifications are beyond its scope. We determined glossary structure and interactive user interface were secondary to improving the performance of the current state-of-the-art in domain-specific keyword detection and definition generation.

6.1 Module 1

Most settings are unavailable via the user interface. One must directly modify the code to perform tasks such as setting custom chinking configurations, increasing training epochs etc. In future GloGen versions we could integrate essential settings into a JSON configuration file.

We chose not to test other SimpleTransformer models within our limited timeframe. Experimenting with different models such as GPT-2 or DistilBERT could improve training time and/or prediction accuracy.

Preprocessing data prior to training is currently bottlenecked by 'keyword pooling'. This mechanism means every word in every text must be checked against every keyword from the entire corpus. This system prevents inconsistency in Wikipedia author hyperlinks from affecting training. It introduces, however, exponentially greater processing time for every additional page. This issue can be resolved via efficient data structures.

6.2 Module 2

The current version of GloGen is unable to source candidate definitions from multiple platforms. In future iterations we would like to add platforms such as Wordnik.

GloGen's current structure is over-complicated and inefficient in some areas, meaning longer texts can take hours for glossary generation. This is because our early development process has heavily prioritised accuracy.

GloGen is still reliant on chinking to remove certain predicted keywords. Improving the BERT training dataset could eliminate the need for this.

GloGen only accepts .txt input at present. We could expand this to include PDF and other file formats.

7 CONCLUSION

GloGen's domain-specific keyword extraction machine learning (Module 1) produces results we can genuinely imagine in an end-system, especially when its predictions are cross-checked by a constituency parser.

Where GloGen has potential for future development is its definition generation (Module 2). While definition prediction is rather performative at approximately 80% accuracy, the resulting glossary lacks the cohesion of human-generated text, which better considers context and interconnection between concepts. The antidote to this is deriving definitions from the original text, rather than external sources such as Wiktionary. Hence, we predict the future of automatic glossary generation lies in the burgeoning field of abstractive text summarisation.

REFERENCES

[1] [n.d.]. 7+ Million Company Dataset. https://www.kaggle.com/peopledatalabssf/free-7-million-company-dataset. Accessed: 2021-03-12.

[2] [n.d.]. English Word Frequency. https://www.kaggle.com/rtatman/english-word-frequency. Accessed: 2021-03-12.

[3] [n.d.]. ibatra/BERT-Keyword-Extractor. https://github.com/ibatra/BERT-Keyword-Extractor. Accessed: 2021-03-12.

[4] [n.d.]. names-dataset 1.9.1. https://pypi.org/project/names-dataset/. Accessed: 2021-03-12.

[5] [n.d.]. NER Data Formats. https://simpletransformers.ai/docs/ner-data-formats/. Accessed: 2021-03-12.

[6] [n.d.]. semantic-text-similarity 1.0.3. https://pypi.org/project/semantic-text-similarity/. Accessed: 2021-03-12.

[7] [n.d.]. Statistics. https://en.wiktionary.org/wiki/Special:Statistics?action=raw. Accessed: 2021-03-12.

[8] [n.d.]. Suyash458/WiktionaryParser. https://github.com/Suyash458/WiktionaryParser. Accessed: 2021-03-12.

[9] [n.d.]. Terms that have different meanings for scientists and the public. https://docs.google.com/spreadsheets/d/1eEBFGRO1UgA6OYoUF9XgRpwgXdliWIYB4J07qElEv-Y/edit?pli=1#gid=0. Accessed: 2021-03-12.

[10] [n.d.]. Wikipedia:Contents/Categories. https://en.wikipedia.org/wiki/Wikipedia:Contents/Categories. Accessed: 2021-03-12.

[11] [n.d.]. Wikipedia:Size of Wikipedia. https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia. Accessed: 2021-03-12.

[12] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alipio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences* 509 (2020), 257–289.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[14] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1262–1273.

[15] Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL interactive poster and demonstration sessions*. 170–173.

[16] Suhan Pan, Zhiqiang Li, and Juan Dai. 2019. An improved TextRank keywords extraction algorithm. In *Proceedings of the ACM Turing Celebration Conference-China*. 1–7.

[17] Jinal Tandel, Kinjal Mistree, and Parth Shah. 2019. A review on neural network based abstractive text summarization models. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. IEEE, 1–4.

[18] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*. PMLR, 11328–11339.

[19] Yong Zhang, Yang Fang, and Xiao Weidong. 2017. Deep keyphrase generation with a convolutional sequence to sequence model. In *2017 4th International Conference on Systems and Informatics (ICSAI)*. IEEE, 1477–1485.