SPRING Goes Online: End-to-End AMR Parsing and Generation

Rexhina Blloshmi
Sapienza University of Rome
blloshmi@di.uniroma1.it

Michele Bevilacqua
Sapienza University of Rome
bevilacqua@di.uniroma1.it

Edoardo Fabiano
Sapienza University of Rome
edoardo10x@gmail.com

Valentina Caruso
Babelscape, Italy
caruso@babelscape.com

Roberto Navigli
Sapienza University of Rome
navigli@diag.uniroma1.it

Abstract

In this paper we present SPRING Online Services, a Web interface and RESTful APIs for our state-of-the-art AMR parsing and generation system, SPRING (Symmetric PaRsIng aNd Generation). The Web interface has been developed to be easily used by the Natural Language Processing community, as well as by the general public. It provides, among other things, a highly interactive visualization platform and a feedback mechanism to obtain user suggestions for further improvements of the system's output. Moreover, our RESTful APIs enable easy integration of SPRING in downstream applications where AMR structures are needed. Finally, we make SPRING Online Services freely available at http://nlp.uniroma1.it/spring and, in addition, we release extra model checkpoints to be used with the original SPRING Python code.[1]

1 Introduction

Abstract Meaning Representation (Banarescu et al., 2013, AMR) is a popular formalism for representing the semantics of natural language in a readable and hierarchical way. AMR pairs English sentences with graph-based logical formulas which are easily accessible by both humans and machines, while abstracting away from many syntactic variations. Because of the formalism's ambition to be comprehensive, AMR graphs are complex objects that require a parser – an automatic algorithm that transduces a natural language utterance into an AMR graph – to subsume multiple traditional Natural Language Processing tasks: Word Sense Disambiguation (Bevilacqua et al., 2021b; Barba et al., 2021), Semantic Role Labeling (Màrquez et al., 2008; Conia et al., 2021; Blloshmi et al., 2021), Named Entity Recognition (Yadav and Bethard, 2018), Entity Linking (Ling et al., 2015; Tedeschi et al., 2021), and Coreference Resolution (Kobayashi and Ng, 2020). Owing to this complexity, AMR parsing, as well as its specular counterpart, i.e., AMR generation, are hard to solve. However, the richness of the information included in AMR graphs, as well as their obvious applications as an interface between human and machines, make both AMR parsing and generation very rewarding problems to solve. As a matter of fact, AMR has been successfully applied to diverse downstream applications, such as Machine Translation (Song et al., 2019), Text Summarization (Hardy and Vlachos, 2018; Liao et al., 2018), Human-Robot Interaction (Bonial et al., 2020a), Information Extraction (Rao et al., 2017) and, more recently, Question Answering (Lim et al., 2020; Bonial et al., 2020b; Kapanipathi et al., 2021). However, since AMR graphs for such applications are obtained automatically through an AMR parser, the benefits of AMR integration are highly correlated with the performance of the underlying parser across various data distributions and domains.

In recent years, AMR parsing and generation models have become more reliable than they used to be, thanks to both the availability of pretrained language models (Devlin et al., 2019; Lewis et al., 2020) and the continuous improvements in the AMR-specific model architectures (Zhou et al., 2020; Cai and Lam, 2020; Fernandez Astudillo et al., 2020). However, most of the existing models make use of cumbersome, data-specific techniques and components which not only limit the out-of-distribution generalizability, but also make it difficult to integrate such models in the pipeline of downstream applications. In our recent paper, SPRING (Bevilacqua et al., 2021a), we proposed a solution through a simple, end-to-end approach with no heavy inbuilt data processing assumptions. Our model achieved unprecedented performance in AMR parsing and generation, both in- and out-of-distribution.

To make SPRING accessible to the community, thereby lowering the entry point to AMR applica-

[1] https://github.com/SapienzaNLP/spring

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 134–142

tion research, we present SPRING Online Services which include:

- a Web interface to easily produce and visualize an AMR graph for a given sentence and, vice versa, a sentence for a given AMR graph in PENMAN (Goodman, 2020) notation.
- RESTful APIs to programmatically request AMR parsing and generation services.
- a bidirectional SPRING model also trained on Bio-AMR, resulting in much stronger performances for biomedical applications.
- a feedback mechanism which allows users to submit modifications to the system's outputs – aided by the visualization – which we collect to enable future enhancements of AMR systems using active learning (Settles, 2009).

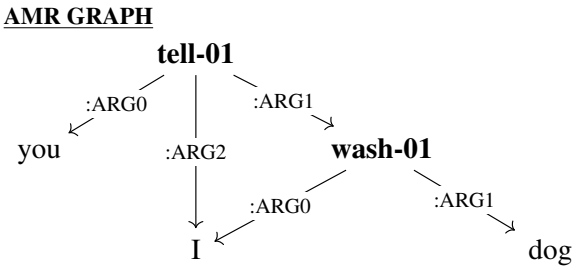
2 SPRING

In this Section we revisit the details of SPRING as in Bevilacqua et al. (2021a) along with the alterations we employ for this demonstration.

2.1 Task Formulation

SPRING is a simple sequence-to-sequence model that operates either as a *parser*, aiming to produce a linearized AMR graph given a sentence, or as a data-to-text *generator*, generating a sentence from an input linearized AMR graph. Formally, a sentence is represented as a sequence of tokens $\mathbf{s} = \langle \text{BOS}, w_1, w_2, \ldots, w_n, \text{EOS} \rangle$ where each word w_i belongs to the vocabulary V, and $\text{BOS}, \text{EOS} \in V$ are special beginning-of-sentence and end-of-sentence tokens, respectively. For example, the sentence *You told me to wash the dog* is represented as ⟨BOS,'You', 'told', 'me', 'to', 'wash', 'the', 'dog', EOS⟩. Similarly, a linearized graph is also a sequence $\mathbf{g} = \langle \text{BOS}, g_1, g_2, \ldots, g_m, \text{EOS} \rangle$, where $g_i \in V$. The graph of the aforementioned sentence is shown in Figure 1. Note that both sentence and graph tokens are drawn from the same vocabulary.

SPRING is at its heart a function P_θ (with θ being the parameters) that takes as input a source string σ in $V^* = \bigcup_{i=1}^{\infty} V^i$ and a partial target string $\tau \in V^*$. Then P_θ outputs a next-token probability distribution over V. Applying this basic function repeatedly, we can assign a probability (P^*) to any string of tokens given another one by factorising it in a left-to-right way as a product of conditional probabilities. This can be applied both to the parsing (by using $\mathbf{s}$ as σ, and the progressively built linearization $\mathbf{g}$ as τ; Eq. 1) and generation (exchanging σ and τ; Eq. 2):



SNT You told me to wash the dog

DFS
```
( <R0> tell-01 :ARG0 ( <R1> you )
:ARG1 ( <R3> wash-01 :ARG0 <R2> :ARG1
( <R4> dog ) ) :ARG2 ( <R2> i ) )
```

Figure 1: The AMR graph for the sentence (SNT) "You told me to wash the dog." with its DFS linearization.

$$P^*_\theta(\mathbf{g}|\mathbf{s}) = \prod_{i=1}^{m+1} P_\theta(g_i \mid \tau = \mathbf{g}_{0:i-1}, \sigma = \mathbf{s}) \quad (1)$$

$$P^*_\theta(\mathbf{s}|\mathbf{g}) = \prod_{i=1}^{n+1} P_\theta(s_i \mid \tau = \mathbf{s}_{0:i-1}, \sigma = \mathbf{g}) \quad (2)$$

To train the model we optimize the parameters to minimize, with mini-batch gradient descent, the so-called negative log likelihood $\mathcal{L}_\theta$ (the negative log conditional probability) over a dataset $\mathcal{D}$ collecting sentence-graph pairs, both for parsing ($\mathcal{L}^{\text{PAR}}_{\theta^{(1)}}$) and generation ($\mathcal{L}^{\text{GEN}}_{\theta^{(2)}}$):

$$\begin{aligned} &\underset{\theta^{(1)},\theta^{(2)}}{\text{argmin}} \quad \mathcal{L}^{\text{PAR}}_{\theta^{(1)}}(\mathcal{D}) + \mathcal{L}^{\text{GEN}}_{\theta^{(2)}}(\mathcal{D}) = \\ &\underset{\theta^{(1)},\theta^{(2)}}{\text{argmin}} \; - \sum_{\langle \mathbf{s},\mathbf{g} \rangle \in \mathcal{D}} \log P^*_{\theta^{(1)}}(\mathbf{g}|\mathbf{s}) + \log P^*_{\theta^{(2)}}(\mathbf{s}|\mathbf{g}) \end{aligned} \quad (3)$$

Note that when $\theta^{(1)}$ is different from $\theta^{(2)}$, the two objective terms are optimized separately. Instead, when we enforce $\theta^{(1)} = \theta^{(2)}$ we have a model that is not only symmetric, but can also perform both AMR parsing and generation at the same time. As we will see, this results in negligible performance drops compared to the disjoint models that we presented in Bevilacqua et al. (2021a).

Once we have the trained model, the predicted output is the string ending in EOS with the highest probability in P^*_θ. Unfortunately, finding this optimal string is intractable when $|V|$ is large; in practice, however, we can perform an approximate decoding with histogram beam search.

2.2 Architecture

The SPRING model is based on the Transformer architecture (Vaswani et al., 2017), a sequence-to-sequence neural network that, briefly, i) uses attention instead of recurrence to encode sequences, ii) is made up of an encoder module that embeds σ, and a decoder that, based on both the encoder output and τ, produces the final distribution output. Key to the high performances of SPRING is the fact that its parameters are not randomly initialized, but, instead, are adopted from those of a large pre-trained encoder-decoder model, i.e., BART (Lewis et al., 2020). Owing to this, SPRING can exploit the extensive knowledge BART encompasses, gained through optimization on large amounts of raw text with an unsupervised denoising objective.

2.3 Linearization

As we have mentioned, the bare SPRING model can translate *from* and *into* linearized AMR graphs. PENMAN, i.e., the format that is used to distribute the AMR meaning bank, is an example of a linearization. In Bevilacqua et al. (2021a) we experimented with different fully graph-isomorphic linearization techniques. The linearization that worked best was the one based on the Depth-First Search (DFS) graph traversal algorithm, enhanced with the use of special tokens to represent the variables, e.g., `<R0>`, `<R1>`, ..., `<Rn>` (Figure 1). Thus, we use the DFS-based linearization here.

One problem when performing parsing is that, since we do not enforce constraints in decoding, the predicted linearization may not be readable back into a valid AMR. In practice, the outputs are almost always valid, or can be made so with little modification. Thus, in parsing only, we perform light, non content-modifying postprocessing, mainly to ensure the validity of the linearization produced, e.g., restoring parenthesis parity and removing duplicate edges. Differently from Bevilacqua et al. (2021a), here we do not employ a third-party Entity Linker so as to avoid response delay.

2.4 Vocabulary

We modify the BART vocabulary in order to make it suitable for AMR-specific concepts (e.g. `amr-unkwown`, `date-entity`), frames (e.g. `say-01`) and relations (e.g. `:ARG1`, `:time`), as well as special pointer tokens used in the DFS linearization. The final vocabulary V is the union of the original BART vocabulary and our additions. Finally, we adjust the tokenization rules so that they do not split AMR additional tokens into multiple sub-words and adjust the encoder and decoder embedding matrices to include the new symbols. To this end, we add a vector for each newly added token which we initialize as the average of the vectors of the sub-word constituents. This is useful for obtaining compact sequences of tokens, allowing for faster decoding and response time of the SPRING Online Services.

3 SPRING Online Services

Here we describe the functionalities of the Web interface (Section 3.1) and those of the RESTful APIs (Section 3.2). We further provide the architectural details and libraries used in Appendix A.

3.1 Web Interface

The main functionalities of the Web interface include switching between *parsing* and *generation* modalities, visual inspection of SPRING results view and the feedback mechanism we develop to enable users to validate SPRING predictions.

3.1.1 Modality Selector

The modality can be set on the initial homepage by choosing `Text` or `PENMAN` from the Tab menu, with `Text` being the default option. When the `Text` option is chosen, the user is required to provide a plaintext sentence and they will then be redirected to the SPRING *parser* Results View (shown in Figure 2). On the other hand, when the `PENMAN` option is chosen, the user is required to type or copy a valid AMR graph in PENMAN notation. In the case when the PENMAN provided is valid, the user is redirected to the SPRING *generator* Results View. Otherwise, when the graph is not valid, the user is notified by a warning which points to the error line number of the PENMAN.

3.1.2 SPRING Results View

The Results View is similar for both parsing and generation, and we only exchange the query (input) box and the result (output) box.

A. Query box. As in the Modality Selector phase, also here, in the *parsing* modality the query box takes as input a plaintext sentence as input, while in *generation* the query box requires the input to be a valid PENMAN. A user can parse or generate from different inputs in this view while remaining in the same modality. To switch from

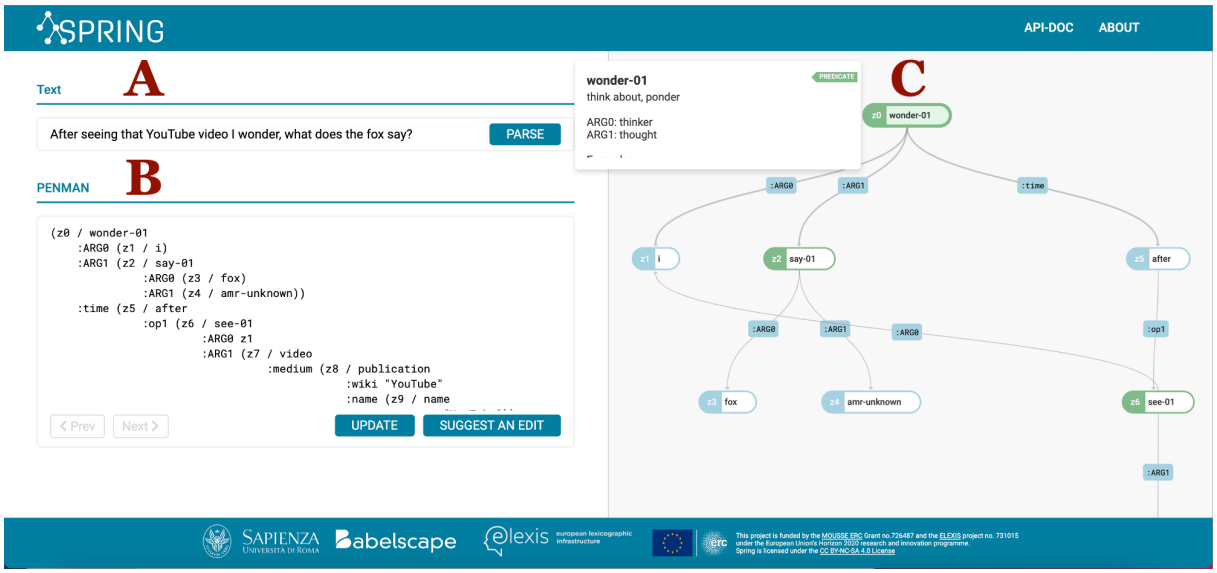


Figure 2: User interface of the SPRING *parser* Results View when the English sentence "After seeing that YouTube video I wonder, what does the fox say?" is typed as input.

parsing to *generation* or vice versa, the user should go back to the initial homepage.

B. Result box. When parsing a sentence, the Result box will be filled with the predicted graph in PENMAN format. This box is editable to enable user feedback (see Section 3.1.3). When *generating* from an AMR graph, the Result box shows the generated sentence which can also be modified by the user and submitted to the feedback system.

C. AMR view panel. This is a key component of the Results View, which visualizes an AMR as a hierarchical graph with labeled nodes and labeled edges. We devise a custom node and edge layout meant to enhance readability even in the case of big graphs with a lot of coreference edges. For example, there might be overlapping edges, edge labels or nodes in the graph. To increase visibility, the user can click/hover on an edge or edge label, and it will be highlighted and brought to the foreground. The same applies to nodes, and in addition, clicking/hovering over nodes will also highlight and bring to the foreground every incoming and outgoing edge, thus identifying all the local relations of a concept. The graph view is resizeable in order to better handle big AMR graphs, and the user is also able to zoom in/out for ease of reading. There are 4 types of node, indicated by different colors, comprising: i) predicate concept nodes, ii) non-predicate concept nodes, iii) constant nodes and iv) wiki nodes. Both predicate and non-predicate nodes are labeled with a variable name (in the highlighted corner) and the concept they represent. The variable makes it easy to locate the node in the PENMAN box on the left Panel.

Futhermore, both predicate and wiki nodes are associated with an `onhover/onclick` tooltip box that further defines them. The tooltip associated with the wiki node contains information taken from the corresponding BabelNet[2] (Navigli and Ponzetto, 2010; Navigli et al., 2021) concept, displaying a short entity description and image (when applicable), also redirecting the user to the corresponding BabelNet page when clicking on it. This choice is motivated by the fact that BabelNet concepts function as a hub of information beyond that of Wikipedia, which paves the way for future integration of other resources in AMR. The tooltip of the predicate node, instead, provides details on the predicate definition and arguments taken from the PropBank framesets (Palmer et al., 2005). In addition, we display an example sentence containing the predicate in the specified sense. The user is redirected to the PropBank predicate page when clicking the tooltip. We mean the extra information shown by the tooltip component to be useful for the user to identify potential parsing mistakes in the output of the system, and ideally to use the provided feedback mechanism to suggest corrections.

3.1.3 Feedback Mechanism

One key functionality of SPRING Online Services that requires user interaction is the Feedback Mech-

[2]Version 5.0.

anism. It is included in both parsing and generation modalities. With this feature, we aim to obtain a manual validation of SPRING output graphs or sentences, aided by the visualization. More specifically, when a user recognizes a mistake of the SPRING *parser*, including both missing or extra nodes and edges, or wrongly labeled ones, they are allowed to suggest modifications. In SPRING *parser* modality, multiple modifications are allowed in the left-panel PENMAN box, which are updated simultaneously in the right AMR view panel when the `UPDATE` button is pressed, and a user can then navigate through their own modifications by means of the `Prev` and `Next` buttons. To submit a final modification request, a user is provided with the `SUGGEST AN EDIT` button. The modifications are accepted if they lead to a correctly-formed graph. When this is the case, we save the modification request in a database for further validation. In contrast, when a mistake is found the user is warned about the line in PENMAN where it occurs. In the SPRING *generator* instead, only the predicted sentence is allowed to be modified, assuming that the input graph by the user is correct and does not need further modification. If this is not the case, the user can query the system with another AMR to obtain a new result. This feedback mechanism paves the way to future advancements in the field:

- enabling the use of active learning for improving system performance;
- collecting human validated SPRING output which can be further used as synthetic data for enhancing AMR systems;
- providing evidence of common SPRING mistakes which can aid studies on interpretation and reinforcement of AMR systems' knowledge.

Since data collection requires time and considerable interaction of users with our services, we leave the exploration of methods for including such data in AMR tasks as future work. Moreover, we plan to release the accumulated data periodically and on-request to the community.

3.2 RESTful APIs

The RESTful APIs we provide can be used effectively to query the SPRING services programmatically. Our APIs are simple and, differently from our Web interface, do not allow modification requests of the SPRING output. The APIs can be accessed through GET or POST requests. In fact, the APIs consist of two endpoints, namely, `/api/text-to-amr` and `/api/amr-to-text`, to parse into or generate from an AMR graph, respectively. The former requires a `sentence` string parameter and the output is a JSON object containing the PENMAN graph, while the latter expects a valid string serialized PENMAN graph, and the response is a JSON object containing the sentence. To ease the usage of the RESTful APIs, the full documentation is accessible through the SPRING Web interface, i.e., `API-Doc` from the header menu bar.

4 Evaluation

For the purposes of this demo, we examine different variants of SPRING to ensure: i) high performance, ii) high generalizability across domains, and iii) efficient and light SPRING Online Services.

Datasets. To deal with i) and ii), we perform experiments with the AMR 3.0 (LDC2020T02[3]) benchmark – currently the largest AMR-annotated corpus which includes and corrects both of its previous inferior-sized versions, i.e., AMR 2.0 and AMR 1.0. In addition to this, motivated by AMR-based approaches in biomedical applications (Rao et al., 2017; Bonial et al., 2020b), we jointly train and evaluate SPRING in the Bio-AMR[4] corpus (May and Priyadarshi, 2017) as well.

Systems. While Bevilacqua et al. (2021a) train one specular model for each of the AMR tasks (henceforth $\text{SPRING}_{\text{uni}}$, denoting unidirectional), to satisfy the point iii) above, we train a version of SPRING that handles both AMR parsing and generation with the same model (henceforth $\text{SPRING}_{\text{bi}}$, denoting bidirectional). This allows us to load into memory only one model to perform both tasks, thus decreasing the potential overload of the server where the demo resides, as well as enabling lower memory footprint for users employing SPRING with our Python code. To train SPRING variants, we employ the same hyperparameters as in Bevilacqua et al. (2021a). In addition, we summarize the state-of-the-art systems on AMR 3.0.

[3] catalog.ldc.upenn.edu/LDC2020T02
[4] amr.isi.edu/download.html

Parsing	Lyu et al. (2020)	75.8
	Zhou et al. (2021)	81.2
	SPRING (Bevilacqua et al., 2021a)	**83.0**
Generation	Zhang et al. (2020)	34.3
	T5 Fine-Tune (Ribeiro et al., 2021)	41.6
	STRUCTADAPT-RGCN (Ribeiro et al., 2021)	**48.0**
	SPRING (Bevilacqua et al., 2021a)	44.9

Table 1: Comparison with literature on AMR 3.0.

			AMR 3.0		Bio-AMR	
		Train dataset	Dev	Test	Dev	Test
Parsing	$\text{SPRING}_{\text{uni}}$	AMR 3.0	83.9	82.6	60.6	60.6
	$\text{SPRING}_{\text{bi}}$	AMR 3.0	83.6	82.3	60.5	59.2
	$\text{SPRING}_{\text{uni}}$	Bio+AMR 3.0	83.9	82.5	**80.0**	80.1
	$\text{SPRING}_{\text{bi}}$	Bio+AMR 3.0	**84.1**	**82.7**	79.5	**80.2**
Generation	$\text{SPRING}_{\text{uni}}$	AMR 3.0	45.0	44.9	22.9	19.4
	$\text{SPRING}_{\text{bi}}$	AMR 3.0	43.9	44.5	21.1	17.1
	$\text{SPRING}_{\text{uni}}$	Bio+AMR 3.0	**45.3**	**45.7**	**39.5**	**43.5**
	$\text{SPRING}_{\text{bi}}$	Bio+AMR 3.0	44.3	45.0	38.5	42.0

Table 2: SPRING variants in AMR 3.0 and Bio-AMR.

Results. We report Smatch (Cai and Knight, 2013) and BLEU (Papineni et al., 2002) scores for AMR parsing and generation, respectively. In Table 1 we summarize the performances of recent systems in the literature on the AMR 3.0 parsing and generation tasks. In *parsing*, SPRING achieves the highest results across the board. In fact, we note that Zhou et al. (2021) was published after Bevilacqua et al. (2021a), yet SPRING remains the best-performing parser in the literature to date. In *generation*, instead, SPRING attains considerably higher results than Zhang et al. (2020) and T5 Fine-Tune (Ribeiro et al., 2021) models. In fact, while the latter has a comparable architecture to that of SPRING due to its use of the pretrained sequence-to-sequence T5 model (Raffel et al., 2019), SPRING nevertheless outperforms it by 3.3 BLEU points. SPRING obtains lower results than the recent STRUCTADAPT-RGCN (Ribeiro et al., 2021) model, which, however, achieved those results at the expense of a more complex architecture with a higher number of parameters than SPRING. In Table 2 we report the performance of SPRING variants, i.e., $\text{SPRING}_{\text{uni}}$ and $\text{SPRING}_{\text{bi}}$, trained on AMR 3.0 or on the concatenation of Bio-AMR and AMR 3.0 (Bio+AMR 3.0) and when evaluated in development and test splits of each. Notice that the results of $\text{SPRING}_{\text{uni}}$ in AMR 3.0 parsing are different from those reported in Table 1, since here, as we recall from Section 2.3, we do not perform Entity Linking in postprocessing for the purpose of simplicity. Firstly, SPRING models trained on Bio+AMR 3.0 achieve the highest results overall. Then, $\text{SPRING}_{\text{bi}}$ performs on a par with or slightly worse than $\text{SPRING}_{\text{uni}}$ in parsing and generation, respectively. We choose the best model for the SPRING Online Services based on the Smatch score on the development set of AMR 3.0, i.e, $\text{SPRING}_{\text{bi}}$ trained on Bio+AMR 3.0 for both parsing and generation jointly. This model allows for the achievement of all the goals we set at the beginning of this Section: performance, generalizability and efficiency. Furthermore, we release the additional model checkpoints to be used with the original SPRING Python code, available at `https://github.com/SapienzaNLP/spring`.

5 Related Work

With a view to demonstrating the progress made in AMR, over the years different Web services for state-of-the-art AMR systems (Konstas et al., 2017; Damonte et al., 2017; Damonte and Cohen, 2019) have been developed. Similarly to SPRING, Konstas et al. (2017), proposed an encoder-decoder system to perform both parsing and generation by relying on data augmentation techniques. This system is associated with a demo[5] to only parse into or generate from AMR and does not provide extra functionalities, RESTful APIs, or any interaction with the users. Similarly, Damonte et al. (2017) and Damonte and Cohen (2019) do not provide other functionalities in their demos beside parsing (Damonte et al., 2017, AMREager)[6] and generation (Damonte and Cohen, 2019, AMRGen)[7]. However, SPRING outperforms the aforementioned systems by more than 20 points in both, Smatch for AMR parsing and BLEU for AMR generation. In addition, through SPRING Online Services we provide a highly-interactive Web interface, RESTful APIs, and the feedback mechanism.

6 Conclusion

With this paper we make available SPRING Online Services, with which we bring state-of-the-art AMR systems into the hands of the community, providing a highly interactive interface and easily integrable APIs. Our SPRING system ob-

[5] Inactive link: www.ikonstas.net/index.php?page=demos
[6] AMREager: bollin.inf.ed.ac.uk/amreager.html
[7] AMRGen: bollin.inf.ed.ac.uk/amrgen.html

tains results in the same ballpark as the state of the art in both AMR-to-Text and Text-to-AMR, using the same weights for both. Moreover, our model obtains very strong results in the biomedical domain, providing a powerful tool for building applications. In the future, we intend to extend the services across languages following techniques as in Blloshmi et al. (2020) and Procopio et al. (2021). We make SPRING Online Services available at `http://nlp.uniroma1.it/spring`.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487, the ELEXIS project No. 731015 and the European Language Grid project No. 825627 (Universal Semantic Annotator, USeA) under the European Union's Horizon 2020 research and innovation programme.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Edoardo Barba, Luigi Procopio, and Roberto Navigli. 2021. ConSeC: Word Sense Disambiguation as Continuous Sense Comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021a. One SPRING to Rule Them Both: Symmetric AMR Semantic Parsing and Generation without a Complex Pipeline. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12564–12573.

Michele Bevilacqua, Tommaso Pasini, Alessandro Raganato, and Roberto Navigli. 2021b. Recent trends in word sense disambiguation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4330–4338. ijcai.org.

Rexhina Blloshmi, Simone Conia, Rocco Tripodi, and Roberto Navigli. 2021. Generating senses and roles: An end-to-end model for dependency- and span-based semantic role labeling. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 3786–3793. ijcai.org.

Rexhina Blloshmi, Rocco Tripodi, and Roberto Navigli. 2020. XL-AMR: Enabling cross-lingual AMR parsing with transfer learning techniques. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2487–2500, Online. Association for Computational Linguistics.

Claire Bonial, Lucia Donatelli, Mitchell Abrams, Stephanie M. Lukin, Stephen Tratz, Matthew Marge, Ron Artstein, David Traum, and Clare Voss. 2020a. Dialogue-AMR: Abstract Meaning Representation for dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 684–695, Marseille, France. European Language Resources Association.

Claire Bonial, Stephanie M. Lukin, David Doughty, Steven Hill, and Clare Voss. 2020b. InfoForager: Leveraging semantic search with AMR for COVID-19 research. In *Proceedings of the Second International Workshop on Designing Meaning Representations*, pages 67–77, Barcelona Spain (online). Association for Computational Linguistics.

Deng Cai and Wai Lam. 2020. AMR parsing via graph-sequence iterative inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1290–1301, Online. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Simone Conia, Andrea Bacciu, and Roberto Navigli. 2021. Unifying cross-lingual semantic role labeling with heterogeneous linguistic resources. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–351, Online. Association for Computational Linguistics.

Marco Damonte and Shay B. Cohen. 2019. Structural neural encoders for AMR-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3649–3658, Minneapolis, Minnesota. Association for Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for Abstract Mean-

ing Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1001–1007, Online. Association for Computational Linguistics.

Michael Wayne Goodman. 2020. Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 312–319, Online. Association for Computational Linguistics.

Hardy Hardy and Andreas Vlachos. 2018. Guided neural language generation for abstractive summarization using Abstract Meaning Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 768–773, Brussels, Belgium. Association for Computational Linguistics.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G P Shrivatsa Bhargav, and Mo Yu. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Hideo Kobayashi and Vincent Ng. 2020. Bridging resolution: A survey of the state of the art. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3708–3721, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract Meaning Representation for multi-document summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1178–1190, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jungwoo Lim, Dongsuk Oh, Yoonna Jang, Kisu Yang, and Heuiseok Lim. 2020. I know what you asked: Graph path learning using AMR for commonsense reasoning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2459–2471, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Chunchuan Lyu, Shay B. Cohen, and Ivan Titov. 2020. A differentiable relaxation of graph segmentation and alignment for AMR parsing. *Arxiv preprint*, abs/2010.12676.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Special issue introduction: Semantic role labeling: An introduction to the special issue. *Computational Linguistics*, 34(2):145–159.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract Meaning Representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Vancouver, Canada. Association for Computational Linguistics.

Roberto Navigli, Michele Bevilacqua, Simone Conia, Dario Montagnini, and Francesco Cecconi. 2021. Ten years of babelnet: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4559–4567. ijcai.org.

Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*,

pages 216–225, Uppsala, Sweden. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Luigi Procopio, Rocco Tripodi, and Roberto Navigli. 2021. SGL: Speaking the graph languages of semantic parsing via multilingual translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 325–337, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *Arxiv preprint*, abs/1910.10683.

Sudha Rao, Daniel Marcu, Kevin Knight, and Hal Daumé III. 2017. Biomedical event extraction using Abstract Meaning Representation. In *BioNLP 2017*, pages 126–135, Vancouver, Canada,. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Yue Zhang, and Iryna Gurevych. 2021. Structural adapters in pretrained language models for amr-to-text generation. *Arxiv preprint*, abs/2103.09120.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using AMR. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Simone Tedeschi, Simone Conia, Francesco Cecconi, and Roberto Navigli. 2021. Named Entity Recognition for Entity Linking: What works and what's next. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*, Punta Cana, Dominican Republic.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yan Zhang, Zhijiang Guo, Zhiyang Teng, Wei Lu, Shay B. Cohen, Zuozhu Liu, and Lidong Bing. 2020. Lightweight, dynamic graph convolutional networks for AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2162–2172, Online. Association for Computational Linguistics.

Jiawei Zhou, Tahira Naseem, Ramón Fernandez Astudillo, and Radu Florian. 2021. AMR parsing with action-pointer transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5585–5598, Online. Association for Computational Linguistics.

Qiji Zhou, Yue Zhang, Donghong Ji, and Hao Tang. 2020. AMR parsing with latent structural information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4306–4319, Online. Association for Computational Linguistics.