Subtl.ai at the FinSBD-2 task: Document Structure Identification by Paying Attention

Aman Khullar[1*], **Abhishek Arora**[1*], **Sarath Chandra Pakala**[1*], **Vishnu Ramesh**[1*] and **Manish Shrivastava** [1†]
[1]Subtl.ai, CIE, IIIT Hyderabad
aman.khullar@iiit.ac.in, abhishek.arora@iiit.ac.in, sarath@subtl.ai, vishnu@subtl.ai, manish@subtl.ai

Abstract

This paper presents a methodology submitted to the FinSBD-2 shared task to extract well formed sentences, lists and items from noisy unstructured financial PDF documents in English language. The proposed architecture for document structure identification, is a combination of deep learning and heuristic based approaches. We use two unidirectional Long Short-Term Memory(LSTM) encoders to get the sentence split tokens from the set of all the possible split points. Further, the outputs are passed on to an attention based LSTM network to select only the well formed sentences from all the possible sentences. These outputs are merged to ultimately produce all possible well formed sentences. Apart from the sentences, lists and items are identified using a combination of heuristics which identify patterns in the data. The final F1 score, 0.217 on this task, is obtained by comparing the start and end indices of sentences, lists and items. We have presented another parameter, which is used to evaluate the class coverage by checking the overlap between the predicted and ground truth sentences and obtained an average 40% class coverage score. This metric is more useful for industry researchers who require coverage of the content rather than character level precision. The proposed approach will empower both academia and industry researchers in their effort to handle noisy documents for various NLP tasks by providing a simple, fast and robust approach to identify structure in their documents.

Keywords:
Sentence Boundary Detection, NLP, Deep Learning, LSTM, Sentence overlap, Token classifier, Document structure identification, Attention mechanism

1 Introduction

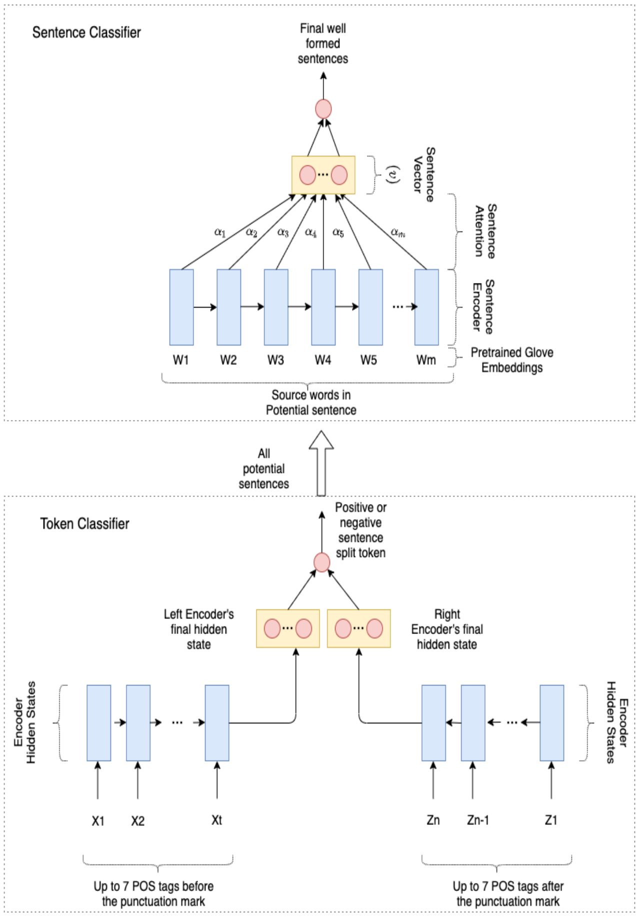
A sentence forms the basic unit of text documents which are used for a wide variety of tasks in Natural Language Processing (NLP) like Named Entity recognition (NER), translation, speech recognition, topic segmentation etc. By definition, ”A sentence is a set of words that is complete in itself, typically containing a subject and predicate, conveying a statement, question, exclamation, or command, and consisting of a main clause and sometimes one or more subordinate clauses.”[1]. It can be clearly realised that it is imperative to clearly demarcate sentence boundaries from noisy text PDF documents so that those sentences in the text can be processed to be used in tasks similar to those mentioned above. Inaccuracy in determination of clear sentence boundaries can lead to misrepresentation of sentence units which can affect the information learnt by networks trained on such noisy data which in turn would affect NLP applications. It is also equally important to demarcate boundaries of list and items to understand the structural and hierarchical features of the document. This disambiguation increments the number of features for the network to learn which drives more accuracy for NLP applications such as identifying procedure and steps.

A sentence boundary is defined from the first character index of the start token of the sentence to the the last character index of the end token of the sentence. Similarly an item is defined in a similar way excluding any unicode bullets, alphanumerics etc. at the starting or ending. A list majorly comprise of a series of items under the same category clubbed together which may include a header sentence.

In the past, this problem in the NLP domain has not gained much importance to disambiguate sentences, list and item boundaries. Heuristics have been long used to identify sentence boundaries which are easy to setup, gives high speed performance but the way each document is setup and formatted without any strict standard rule of formatting, the accuracy of identification takes a hit, therefore, it becomes imperative to develop a robust way to identify such boundaries and have a reliable and clear disambiguation of sentences, items, lists as input data which is used for a wide variety of NLP tasks like sentiment analysis, NER, translation, speech recognition etc. This brings in the need for deep learning models which have gained importance recently which can learn a wide variety of patterns, semantic, contextual, positional information etc.

To solve this challenging task of sentence, list and item boundary detection, we have developed a deep learning model supported by task specific rules. Through this paper,

[*]Equal Contribution. Listing order is random.
[†]Contact Author



Sentence Classifier
Final well formed sentences
Sentence Vector
(c)
Sentence Attention
α_1 α_2 α_3 α_4 α_5 α_m
Sentence Encoder
W1 W2 W3 W4 W5 Wm
Pretrained Glove Embeddings
Source words in Potential sentence
All potential sentences
Token Classifier
Positive or negative sentence split token
Left Encoder's final hidden state
Right Encoder's final hidden state
Encoder Hidden States
X1 X2 Xt
Zn Zn-1 Z1
Up to 7 POS tags before the punctuation mark
Up to 7 POS tags after the punctuation mark

Figure 1: Sentence boundary identification architecture

our main contributions are as follows:

- We present a novel sentence boundary detection architecture which is able to give a competitive performance on noisy PDF documents.
- We propose heuristics to identify lists and items in the noisy PDF document.
- We propose content coverage evaluation metric which evaluates content overlap rather than precise character match.

The paper is structured as follows :- 1) Introduction section presents the problem and the need to disambiguate sentence boundaries while section 2 presents the research findings and development in this field. Section 3 describes the particular task for which the dataset description is given in section 4 and section 5 describes the proposed model to solve this task. The results are presented and discussed in section 6 and section 7 concludes the work which is followed by references.

2 Related Work

Traditionally the task of SBD has been solved using heuristics and rules based on the regular grammar. One of the popular approaches presented by Alembic information extraction [2] built an extensive regular-expression-based approach to solve this problem. There have been other rule-based approaches by [3], [4] and [5]. Palmer et al.[6], however, recognized the shortcomings of the rule based approaches which were problem statement specific and required large manual effort. Hence they developed the Satz system which predicted if any punctuation mark was a sentence split point or not. They were among the first to develop machine learning based approach for solving this problem and since then many machine learning based approaches by [7], [8], [9], [10] and [11] approaches. Recently deep learning tools [12] have been used to solve this problem which have been able to produce state of the art results. Many of these approaches have however been confined to clean texts and have tested their results on WSJ corpus [13] and the Brown Corpus [7]. Azzi et al. [14] presented their solution for detecting sentence boundaries in Noisy text in the financial domain but their solution was limited to detecting sentences and not the lists and items contained within these documents. Our work improves upon this shortcoming to identify the lists and list items inside the noisy PDF documents along with the identification of sentences which can be used by any NLP system in their preprocessing step to get state of the art performance.

3 Task Definition

The goal for the FinSBD-2 2020 Shared Task [15] is to extract well segmented sentences identifying them by their start and end indices along with lists and items in English and French. This task is mainly split into two sub tasks :

1. Extraction of sentences, lists and items depicted by their starting and ending indices.
2. Arranging the lists and items in a hierarchical manner.

This task is provided with financial PDF documents as the training data along with a JSON corresponding to each PDF document. Each JSON consists of raw text extracted from the PDF and disambiguated clearly under each class such as sentence, list, item, item1, item2, item3 and item4 which is represented by their start and end index. Co-ordinates of these boundaries are provided as well for spatial and visual cues.

In this paper we focus on the first subtask which includes to disambugate text into sentence, list and item class in English language

There are two other PDF documents provided with their raw text exracted in a JSON file along with the spatial coordinates. This acts as a testing set on which the final models developed by all participating teams is evaluated. F1 scores are calculated for each class and the average is taken for the three classes (sentence, lists and items) for this subtask.

4 Dataset

The data that has been used was provided as part of the FinSBD-2 2020 challenge which contained 6 noisy financial PDF documents for training and 2 documents used for testing. The data consisted json files for each document with the start and end indices of each class namely - sentences, lists and items, along with the complete document text available as a single string. The coordinates of the starting and the ending characters for each of the elements in the class were also

	Train data	Val data	Test data
#Docs	5	1	2
#Characters	1,322,767	169,546	558,611
#Tokens	290,092	39,816	141,138
#Sentences	7,282	788	2,450
#Lists	207	42	69
#Items	843	268	332
#OOV Tokens	1,079	248	443

Table 1: Corpus statistics. The number of characters, tokens, sentences, lists, items and OOV words have been identified for each document.

provided. From these 6 PDF documents, 5 have been used for training our models while 1 was used for validation and 2 were used for testing. The data statistics for the documents in each of these train, val and test splits have been provided in Table 1. Data preprocessing involved cleaning the leading space characters from sentences and then tokenizing the sentence using Spacy's tokenizer [16]. The data was later processed to get a list of all ':', '\n' and '.' tokens along with a label to identify if the token was a positive split token (if it led to sentence, list or item split) or a negative split token (if it did not lead to sentence, list or item split).

5 Our Models

In this section we describe (1) our token classifier (2) sentence classifier (3) sentence boundary detection algorithm and (4) heuristics to identify sentences, lists and items from the text.

5.1 Token Classifier

Through our examination of the training data we identified that the potential tokens which could act as sentence split tokens were three characters namely - '\n', '.', and ':', which we call as potential sentence split tokens. Each sentence would end with a potential split token, even those sentences which do not have a visible punctuation mark, because in that case, the sentence ends with '\n' token. The token classifier model as depicted in Figure 1 illustrates how our model helps us identify the potential sentence split tokens among all the occurrences of these three tokens.

For every occurrence of any of these potential split tokens, we do the following steps:

(a) Convert the previous 7 tokens and the next 7 tokens into their corresponding POS tags.

(b) Convert these POS tags into their one hot vectors.

(c) Pass the previous 7 one hot POS encodings into a forward directional LSTM [17] with the last timestep corresponding to the token just before the potential split token being classified.

(d) Pass the next 7 one hot POS encodings into a backward directional LSTM with the last timestep corresponding to the token just after the potential split token being classified.

(e) Concatenate the final hidden states of these two LSTM encoders and pass a linear layer over it to classify if the potential split token is a positive sentence split token or not.

Domain specific documents contain several words which are not present in the pretrained open-domain word embeddings. As a result, the POS embeddings helped us overcome the ambiguity caused due to the out of vocabulary words while keeping the model computationally less expensive. The hyperparameter value of 7 has been taken as a large enough value to allow the encoder to understand the flow of the POS sequences, while being small enough to train the model efficiently. If on either sides of the point under consideration, we find another potential split token before the span of 7 tokens then this smaller set of tokens are passed to the encoder. This helps ensuring the independence of potential split tokens from each other and does not allow error to propagate in the case of incorrect prediction. The two unidirectional LSTMs complement each other as the final state of the forward directional LSTM maps the flow of POS encodings from the previous sentence and the backward directional LSTMs final hidden state maps the reverse flow of the POS encodings from the next sentence. These hidden states are then concatenated which helps the model to understand how the flow of POS tags, from both the previous as well as the next sentence, determines the split of a sentence.

5.2 Sentence Classifier

Drawing inspiration from previous work in well formed natural language queries [18], the sentence classifier model identifies if any given sentence is well formed or not. We were able to achieve good results for this sub-task by using an attention based LSTM model [19] as depicted in Figure 1.

This model tokenizes the sentences and then we use the pretrained Glove embeddings to get the embedding of each token present in the vocabulary. At each timestep (t), a word (W_t) is embedded using these pretrained embeddings. An LSTM encoder layer passes over these embeddings. At each timestep, we get a hidden state (h_t) which is followed by an attention layer on top of it:

$$x_t = W_e w_t, t \in [1, m] \quad (1)$$

$$\vec{h_t} = \overrightarrow{LSTM}(x_t), t \in [1, m] \quad (2)$$

$$u_t = \tanh(W_w h_t + b_w) \quad (3)$$

$$\alpha_t = \frac{\exp(u_t{}^T u_w)}{\Sigma_t \alpha_t h_t} \quad (4)$$

$$v = \Sigma_t \alpha_t h_t. \quad (5)$$

Here W_e represents the embedding matrix, x_t represents the embedded word, h_t represents the hidden state of the LSTM encoder at each timestep t, u_t is a word level context vector, α_t is the attention weight to given to each word in the input sentence and v is the final sentence [20] vector which captures the information for that sentence. This is followed by a linear layer to classify if the input sentence is a well formed sentence or not.

This sentence classifier has been used by us in two ways, namely :

(1) Overcome the shortcomings of the token classifier.

(2) Identify the end points for lists.

This helps us facilitate in improving our identification of the sentences, lists and items which has been explained in the next section.

5.3 Sentence Boundary Identification

Through the output of the sentence classifier, it becomes known if the sentence identified by the token classifier is a well formed sentence or not. In case it is a well formed sentence then we keep it, as it is. Otherwise, in case it is not well formed sentence, we merge this sentence with the previous identified sentence and then pass the concatenated sentences to the sentence classifier. In case these concatenated sentences turn out to be well formed, we consider both of these sentences as a single sentence. In case the concatenation with the previous sentence is also not able to form a well formed sentence then we try the same approach with the next sentence from the token classifier. If that also does not yield a well formed sentence, we skip this sentence. The result from this merge algorithm gives us the final set of well formed sentences.

5.4 Lists and Items Boundary Identification

The approach to detect items and thus the aggregated lists was focused around patterns specific to lists. From the training data, it was observed that 90% of items are either an alphanumeric pattern (”1.”, ”a.”) or a Unicode pattern which start with a non-ASCII character to represent bullet points. After using these heuristics to find the start indices of every item, the list were aggregated by identifying:

(a) Co-occurrences of alphanumeric or Unicode class items within a window of 7 sentences.

(b) Incremental co-occurrences of alphanumeric pattern-based items(”1.” followed by ”2.”, or ”a.” followed by ”b.”)

Post this aggregation, the end index of the last item was identified using the sentence classifier. Based on the potential sentence split tokens, text blocks were added one at a time to the last item until the sentence classifier categorised the last item as well-formed. To account for inaccuracies in the sentence classifier, a window limit of 4 was set up, meaning not more than the next 4 text blocks were added to identify the end of the last item, and thus the end of the list.

6 Results and Analysis

6.1 Evaluation Metrics

The official evaluation metric is based on matching the start indices and end indices of sentences generated through the proposed methodology which are matched with the ground truth. finSBD 2020 has provided the start and end indices ground truth of each PDF document. The final F1 score takes an average F1 scores of all documents present in the test set. Another metric which is the sentence coverage is evaluated to illustrate the percentage overlap of the detected sentences with the ground truth sentences. Both the metrics are described below.

Document	Class	Precision	Recall	F1
Document 1	Sentence	0.67	0.62	0.64
	List	0.00	0.00	0.00
	Items	0.00	0.00	0.00
	Average	**0.22**	**0.20**	**0.213**
Document 2	Sentence	0.71	0.62	0.66
	List	0.00	0.00	0.00
	Item	0.00	0.00	0.00
	Average	**0.24**	**0.21**	**0.22**

Table 2: F1 score on 2 test documents

6.1.1 F1 scores

F1 score which is the geometric mean of precision and recall which are defined in eq. (6) and (7). F1 scores depicted in eq. (8) is calculated for each class i.e sentence, list and items is evaluated for each document and a simple average is taken. Predicted list, sentence or item is considered as a true positive if the start and end index generated with the help of the proposed methodology matches exactly with that present in the ground truth else it is treated as a false positive.

$$Precision\ (P) = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{6}$$

$$Recall\ (R) = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{7}$$

$$F1\ Score = 2\ .\frac{P\ X\ R}{P\ +\ R} \tag{8}$$

The results on the 2 test documents provided are shown in table 2.

6.1.2 Sentence Coverage

Sentences coverage is calculated by finding the percentage overlap of the predicted sentence with the sentence present in the ground truth. The formula to find this metric is presented below in eq. (9).

$$SentenceOverlap\,\% = \frac{Len(Common\ substring)}{Len(Ground\ truth\ sentence)} \tag{9}$$

For ex. if the ground truth sentence is ”It is a good day.” and the predicted sentence is ”good day”, the common substring will be ”is good day” which has a length of 11 characters. Length of the ground truth sentence in this case will be 16. The corresponding sentence overlap wll thus be 68.75%.

The percentage overlap is calculated for each predicted sentence and is average over the number of predicted sentences. This is depicted in eq. (10)

$$Avg.\ Overlap\,\% = \frac{\sum Sentence\ Overlap}{Total\ no.\ of\ predicted\ sentences} \tag{10}$$

Using this formula, average overlap for each class including sentence, list and item is calculated. The results are presented in the table 3 for the 2 given test documents.

Document	Class	Overlap *100 (%)
Document 1	Sentence	0.82
	List	0.20
	Items	0.16
	Average	**0.39**
Document 2	Sentence	0.91
	List	0.16
	Item	0.15
	Average	**0.41**

Table 3: Sentence Coverage on 2 documents

6.2 Analysis

The analysis on the 2 test documents on both the evaluation metrics, gives a clear idea that deep learning model as proposed in this paper, outperformed rule-based approaches. The detection of sentences which used deep learning model was able to learn more patterns as compared to the heuristics built to identify lists and items. For industry applications, which is more concerned with accuracy rather than precision, F1 score becomes too strict metric in cases where just start and ending indices are compared to classify a sentence, list or an item as true positive, false positive or false negatives. Average F1 score on the 2 test documents for sentences is 0.65 with precision almost equal to recall indicating the number of false negative and false positives equal showing the dataset had comparable number of positive and negative sentences. The F1 scores does not give an idea for the list and items identified by the proposed methodology as the predicted items and list did not match exactly with the ground truth.

To get a better idea of the accuracy of the predicted list and items, sentence coverage metric is evaluated to show the overlap of the predicted sentences with those present in the ground truth. Sentences show an encouraging metric of an average score of 86.5% on the given set of 2 documents. The list have a average of 18% and items 15.5%. The main difference in the way of identification of sentenced and list or items is the usage of deep learning models in the former and logical heuristics for the latter. This is also due to the reason that each document does not follow a set of standard/universal rule for setting up of list and items which makes their identification tedious.

7 Conclusion and Future Work

This paper proposes a deep learning model which consists of primarily two stages to detect a sentence. In the first stage, positive split points to split the raw text string are found using a two uni-directional LSTMs. The sentences are split at these points and passed through a sentence classifier based on attention LSTMs, which classifies a sentence as well formed or not. Not well formed sentences are passed through a merge algorithm to finally get well formed sentences as output. The F1 metric when tested 2 test documents presents 0.65 as the average F1 whereas the sentence coverage gives an average overlap of 86.5%. This paper also proposes a set of heuristics to identify list and items in an unstructured document. F1 score to evaluate the accuracy of the prediction does not give a holistic picture as it simply matches the start and end index to evaluate. Hence, sentence coverage is used which gives an average of 18% for list and 15.5% on item on 2 documents. This shows the amount of overlap in the prediction when compared with the ground truth.

A lot of work has been done in the domain of NLP to detect sentences in unstructured documents and the accuracy of detection in the proposed model in this paper is encouraging. There is clear scope of improvement in the accuracy of detection of list and items in an unstructured PDF document. A deep learning model can be used to train a model to learn a wide variety of features which can be helpful in detecting list and item which will eventually increase the performance of a wide variety of NLP applications which depend upon this data processing step as input.

8 Acknowledgment

We would like to thank Product Lab and the Language Technologies Research Center (LTRC) at IIIT Hyderabad for providing us with the servers and the necessary compute used to train the models.

References

[1] Grammar-Monster, “Definition of sentence.” https://www.grammar-monster.com/glossary/sentences.htm.

[2] J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain, “Mitre: description of the alembic system used for muc-6,” in *Proceedings of the 6th conference on Message understanding*, pp. 141–155, Association for Computational Linguistics, 1995.

[3] C. Hoffmann, “Automatische disambiguierung von satzgrenzen in einem maschinenlesbaren deutschen korpus,” *Manuscript, University of Trier, Germany*, 1994.

[4] G. Grefenstette and P. Tapanainen, “What is a word, what is a sentence?: problems of tokenisation,” 1994.

[5] A. Mikheev, “Periods, capitalized words, etc.,” *Computational Linguistics*, vol. 28, no. 3, pp. 289–318, 2002.

[6] D. D. Palmer and M. A. Hearst, “Adaptive multilingual sentence boundary disambiguation,” *Computational linguistics*, vol. 23, no. 2, pp. 241–267, 1997.

[7] D. Gillick, “Sentence boundary detection and the problem with the us,” in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pp. 241–244, 2009.

[8] T. Kiss and J. Strunk, “Unsupervised multilingual sentence boundary detection,” *Computational linguistics*, vol. 32, no. 4, pp. 485–525, 2006.

[9] F. Wong and S. Chao, “isentenizer: An incremental sentence boundary classifier,” in *Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering (NLPKE-2010)*, pp. 1–7, IEEE, 2010.

[10] D. F. Wong, L. S. Chao, and X. Zeng, “isentenizer-: Multilingual sentence boundary detection model,” *The Scientific World Journal*, vol. 2014, 2014.

[11] D. Rudrapal, A. Jamatia, K. Chakma, A. Das, and B. Gambäck, "Sentence boundary detection for social media text," 2015.

[12] C. Xu, L. Xie, and X. Xiao, "A bidirectional lstm approach with word embeddings for sentence boundary detection," *Journal of Signal Processing Systems*, vol. 90, no. 7, pp. 1063–1075, 2018.

[13] A. Taylor, M. Marcus, and B. Santorini, "The penn treebank: an overview," in *Treebanks*, pp. 5–22, Springer, 2003.

[14] A. A. Azzi, H. Bouamor, and S. Ferradans, "The finsbd-2019 shared task: Sentence boundary detection in pdf noisy text in the financial domain," in *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, pp. 74–80, 2019.

[15] finSBD 2, "finsbd-2 shared task definition." https://sites.google.com/nlg.csie.ntu.edu.tw/finnlp2020/shared-task-finsbd-2?authuser=0.

[16] E. AI, "Spacy library." https://spacy.io/.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] B. Syed, V. Indurthi, M. Gupta, M. Shrivastava, and V. Varma, "Inductive transfer learning for detection of well-formed natural language search queries," in *European Conference on Information Retrieval*, pp. 45–52, Springer, 2019.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[20] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

[21] H. Abelson, G. J. Sussman, and J. Sussman, *Structure and Interpretation of Computer Programs*. Cambridge, Massachusetts: MIT Press, 1985.

[22] R. Baumgartner, G. Gottlob, and S. Flesca, "Visual information extraction with Lixto," in *Proceedings of the 27th International Conference on Very Large Databases*, (Rome, Italy), pp. 119–128, Morgan Kaufmann, September 2001.

[23] M. Ostendorf, M. Collins, S. Narayanan, D. W. Oard, and L. Vanderwende, "Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

[24] R. J. Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system," *Cognitive Science*, vol. 9, pp. 171–216, April–June 1985.

[25] G. Gottlob, "Complexity results for nonmonotonic logics," *Journal of Logic and Computation*, vol. 2, pp. 397–425, June 1992.

[26] G. Gottlob, N. Leone, and F. Scarcello, "Hypertree decompositions and tractable queries," *Journal of Computer and System Sciences*, vol. 64, pp. 579–627, May 2002.

[27] H. J. Levesque, "Foundations of a functional approach to knowledge representation," *Artificial Intelligence*, vol. 23, pp. 155–212, July 1984.

[28] H. J. Levesque, "A logic of implicit and explicit belief," in *Proceedings of the Fourth National Conference on Artificial Intelligence*, (Austin, Texas), pp. 198–202, American Association for Artificial Intelligence, August 1984.

[29] B. Nebel, "On the compilability and expressive power of propositional planning formalisms," *Journal of Artificial Intelligence Research*, vol. 12, pp. 271–315, 2000.

[30] IJCAI Proceedings, "IJCAI camera ready submission." https://proceedings.ijcai.org/info.