ECAI 2018 - International Conference – 10th Edition
Electronics, Computers and Artificial Intelligence
28 June -30 June, 2018, Iasi, ROMÂNIA

The impact of summarisation on textual entailment - a case study on global warming arguments

Adrian Groza and Oana Maria Popa,
Department of Computer Science,
Technical University of Cluj-Napoca, Romania
adrian.groza@cs.utcluj.ro,oana.popa@isg.cs.utcluj.ro

***Abstract*—The challenging task of recognizing textual entailment aims to check whether the meaning of a smaller text - the hypothesis h - can be inferred from another text T. Current methods interleave natural language processing, machine learning, search and lexical resources. All these instruments pose computational challenges that make textual entailment unfeasible for large texts. Hence, we investigate how textual entailment is affected by text summarization. By summarising the text T we expect a decrease of accuracy, but an increase of computation speed. We aim to assess the expected decrease in accuracy caused by summarisation against time benefits due to smaller text given to entailment machinery. Our results show that the time needed for computing entailment is decreased four times, while the accuracy decreases with two percentages.**

***Index Terms*—textual entailment, text summarisation, corpus of arguments**

I. Introduction

The challenging task of recognizing textual entailment (TE) aims to check if a natural language text T entails a smaller statement h. Current methods use a bag of tricks from natural language processing (NLP), machine learning, search, and lexical knowledge bases.

All these instruments pose computational challenges. NLP has difficulties to build syntactical trees for large texts [1]. Machine learning relies on training pairs that need to be annotated as entailment or non-entailment, often manually [2]. Searching for similarities between T and h is done by replacing words in h with synonyms, antonyms, homonyms or with different paraphrases. The search space is huge, given the branching factor of available relationships among words and given the required backtracking during search [3]. The above complexity imposes restrictions to apply TE on real world texts. To avoid these limitations, before given to the entailment machinery, one option is to summarise the texts.

However, summarisation may impact the effectiveness of textual entailment, as relevant words may be removed. In this paper, we investigate the impact of summarisation on the performance of textual entailment. Our first hypothesis was that the accuracy will decrease, but with better computational time. Hence, we analyse the threshold between accuracy and time, guided by the following research questions:

978-1-5386-4901-5/18 $31.00 © 2018 IEEE

- Q_1: How does the size of the training set impact the performance metrics of textual entailment?
- Q_2: How does the processing time vary before and after the summary?
- Q_3: What is the impact of summaries on textual entailment?

II. Discussion and related work

Textual entailment is a computational challenging task in NLP. Bos et al. advocate in [4] that recognizing entailment bears similarities to Turings test, as access to different knowledge sources and the ability to draw conclusions are among the primary ingredients for an intelligent system. Many NLP tasks do have strong links to entailment. In case of the *summarization* task, the summary should be entailed by the original text. *Paraphrases* can be seen as mutual entailment between a text and its paraphrased version. In *information extraction*, the extracted information should be entailed by the text content, whereas in *machine translation*, the automatic translation should be entailed by the golden standard of human translation.

Recognising textual entailment challenge has started in 2005 as an attempt to promote an abstract generic task that captures major semantic inference needs across applications [5], [6], [7]. Participants are provided with pairs of short texts, which are called text-hypothesis to learn or tune their models. Then, a test dataset is provided to run the systems and the performances are gathered. The collected examples represent a range of different levels of entailment reasoning, based on lexical, syntactic, logical and world knowledge at different levels of difficulty.

Different approaches have been proposed to recognize textual entailment: from unsupervised, language independent methodologies [8], [9], [10] to deep linguistic analyses [11], [12], [13]. Perez et al. have used in [9] the BLEU algorithm [14], that is an unsupervised language-independent approach. Byer et al. have treated in [10] the entailment data as an aligned translation corpus. The GIZA++ toolkit is used in [15] to induce alignment models. However, the alignment scores alone have been proved of small importance for the RTE-1 development data: the predicted entailment has been only slightly above chance. As a consequence, Bayer et al. have introduced in [10] a combination of metrics intended to measure translation quality. Pais et al. have proposed

in[16] an unsupervised, language-independent, threshold free methodology to recognize textual entailment by generality. The Asymmetric InfoSimba similarity measure has been used to assess whether a sentence is more specific or more general than another one.

We also applied textual entailment in the climate change domain in [17], [18]. The chatbot in [17] uses textual entailment machinery and ontologies to identify the best answer for a statement conveyed by a human agent. The work in [18] focuses on including domain knowledge into the searching for entailment or non-entailment. Ontologies in description logics are translated into lexical rules suitable for existing textual entailment algorithms. Difficulties arise both in [17] and [18], when the text to analyses in large. These difficulties have lead us to the current work, in which we try to reduce large texts through summarisation.

Text summarization began in the late sixties, when Luhn [19] and Edmundson [20] started to to produce summaries automatically. Since then, many different techniques have been developed [14]. A point of reference is the work of Jones et al. [21]. Their classification is based on the level of processing that each system performs. From this perspective, summarisation can be characterized as approaching the problem at the surface, entity, or discourse level [10]. Riloff et al. have carried out [22] a review of summarisation in the last decade. Furthermore, Alonso et al. in [4], [23] have gave a general overview of summarization systems, providing also a description of their main features and techniques. One observation is that most systems combine several features instead of using only one. For example, NeATS [15] combines techniques such as sentence position, term frequency, topics signature, whereas MEAD [24] relies on centroid score or overlap with the first sentence.

Attempts to study the influence of textual entailment on summarization have been focused on the evaluation of summaries [25] to determine which candidate summary, among a set of them, better represents the content in the original document depending on whether the summary entails it or not. However, very little effort has been done to consider both fields together to produce extracts. One such exception is the work of Cleuziou et al. in [26], where a summary is generated either directly from the entailment relations within a text, or by extracting the highest scored sentences of a document. The score of each sentence is computed as the number of sentences of the text that are entailed by it.

III. Materials and Method

Our method requires three resources: i) a corpus of textual entailment pairs to build the language model for textual entailment; ii) the Excitement Open Platform tool for computing entailments; and iii) various tools for text summarisation. Fig. 1 shows the interleaving of these resources. We used three summarisers: Classifier4J.Jar, SummaryApp, and CNGL. Training pairs that contain the summarised text are given to textual entailment machinery. The corpus of arguments, the summarsers and the textual entailment method are detailed next:

A. Corpus of arguments on global warming

We use the debate corpus related to climate change from http://users.utcluj.ro∼/agroza/projects/argclime. The corpus contains 142 debate topics with 877 pro and against arguments extracted from three debate sites: *Debatepedia* (http://www.debatepedia.org), *debate.org* (http://www.debate.org), and *For and Against* (http://www.forandagainst.com). The advantage of using debate sites is that arguments are already annotated by their creators with "pro" or "against". We used this corpus of annotated arguments to train the textual entailment machinery. That is, we build a language model for the entailment relationship in the climate change domain.

The corpus is used by the machine learning component of the entailment algorithms to learn entailment and non-entailment relations between sentences. Each argument contains of pairs of texts and hypothesis. The pairs are annotated with two relations: *entailment* or *contradiction (non-entailment)*.

Example 1 (Entailment relation):
T : *Arctic vegetation zones are affected by the climate changes.*
h : *The polar desert is affected by the global warming.*

Example 2 (Non-entailment (contradiction) relation):
T : *Global warming is not caused by human activity..*
h : *Climate change is manmade.*

B. Excitement Open Platform

Excitement Open Platform (EOP) is a generic architecture implementing textual inferences in several languages [27]. The platform provides entailment algorithms and facilitates connection various lexical knowledge bases. The EOP takes $\langle T, h\rangle$ pairs as input and the output is an entailment decision. EOP enacts modularization with pluggable components. The EOP's architecture is formed of: the Linguistic Analysis Pipeline (LAP), the Entailment Decision Algorithm (EDA), and knowledge resources.

The Linguistic Analysis Pipeline (LAP) is a collection of annotation components (i.e. Tokenizing, POS Tagging, Dependency Parsing), where component integration is based on the Apache UIMA framework [28].

The Entailment Decision Algorithm (EDA) computes an entailment decision based on entailment algorithms or knowledge resources. The available algorithms are grouped in three types: transformation-based, edit-distance based, and classification based. For our experiments, we will use the the transformation-based BIUTEE algorithm.

Knowledge resources are crucial to recognize cases where T and h use different textual expressions while preserving entailment [18]. The EOP includes a wide range of knowledge resources (WordNet, Verb Ocean, Wikipedia), but also including lexical and syntactic resources (i.e. paraphrasing

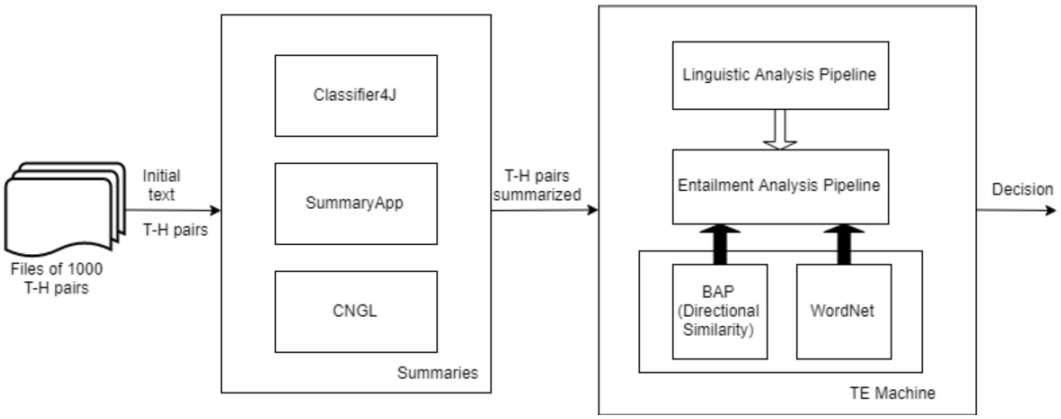

Fig. 1: Our method uses a corpus of arguments, three summarisers and textual entailment machinery.

corpus), where some of them are grabbed from dictionaries, while others are learned automatically.

C. Tools for summarisation

A summary is produced from one or more texts and it contains a significant portion of the information in the original text(s). Hovy et al. have argued that the summary should be no longer than half of the original text(s) [29]. Following the Sparck's approach [21], there are three context factors that influence summaries: input, purpose and output factors. Also, there are two approaches for automatic summarization: extraction and abstraction. Summarisation by extraction consists of selecting a subset of existing phrases in the original text, to form the summary. Summarisation by abstraction builds a semantic internal representation, and then uses natural language-generating techniques to create a summary that is (arguable) closer to what a human could generate. Such a synthesis could contain words that are not explicitly present in the original text.

The following summarisers were used for our experiments: Classifier4j [30], SummaryApp [31], and CNGL summarizer [32], [33].

Classifier4j generates a quality summary compared to other tools, (e.g. MS Word summarizer), based on the following steps: i) deleting HTML, stopwords, etc; ii) sorting unique words by popularity in text; iii) displaying the initial text after the sentences bounds; iv) including in each sentence that first mentions of the most popular word until the required maximum length is met [30].

SummaryApp splits the original text into paragraphs, and then picks the best sentence from each paragraph according to the sentences dictionary [31]. The sentences dictionary calculates a score for each input sentence in a two-steps process: First, the intersection value between each two sentences is stored in a matrix. That is, the text is converted into a fully-connected weighted graph. Each sentence is a node in the graph and the two-dimensional array holds the weight of each edge. Second, an individual score for each sentence is calculated by summing up all its intersections with the other sentences in the text. The intersection function receives two sentences, and returns a score for the intersection between them. We just split each sentence into tokens, count how many common tokens we have, and then we normalize the result with the average length of the two sentences.

CNGL summariser provides a sentence extracted summaryr [32], [33]. There are three stages in the summarization pipeline: i) tokenization: a structurer is used to tokenize and structure the content by sentences and paragraphs; ii) weighting: using the content structure and chosen features weights are assigned to each sentence; iii) aggregation: an aggregator combines the weights from each feature - it may decide to completely discount those negatively weighted, combine them linearly, logarithmically, or in any other fashion. The sentences are then ranked. The summarizer outputs the number of sentences desired according to their score.

IV. Interleaving textual entailment and summarisation

A. Designing experiments

Let T^i_{sum} the text summarised with the sum summariser. The upper index indicates the amount of summarization performed on the initial text T. For instance, T^2_{C4J} states that the C4J summariser was used to summarise T into two sentences. Similarly, $T^{70\%}_{C4J}$ states that C4J tool was used to summarise T with a 70% percent of the number of sentences from the original text.

Our experiments aim to assess the impact of summarisation with respect to time and entailment performance. The top level design of our experiment is:

1) Measuring the running time and entailment performance on initial pairs $\langle T, h\rangle$
2) Measuring the running time and entailment performance on pairs formed by the summarised text and the original hypothesis $\langle T^i_{sum}, h\rangle$.

The variables that we change during experiment are: 1) the summarisation tool, 2) the percentages of summarisation, 3) the size of the training corpus. The results of this experiment

are valuable for cases in which T is large, h is small, and when the corpus contains enough pairs for training.

B. Preprocessing data

We encountered three difficulties during the sentence annotation stage.

First, we observed that the syntactical parser has difficulties to analyse texts larger than 30 sentences. The system fails to identify relative pronouns in such large texts.

Second, a difficulty comes from the *Truth Teller module*, responsible to assess if the sentence is positive or negative. TruthTeller [34] is released as part of Excitement Open Platform (EOP); his output is a CoNLL format with four additional columns to hold the values of annotations: signature, negation uncertainty, clause truth, predicate truth.

T and h are represented by dependency parse trees. Each of graph templates are represented using Stanford dependencies, while adopting a reduced version of POS (part-of-speech) tag set, comprised of: *noun*, *verb*, *adjective*, *adverb*, *preposition*, *determiner*, *pronoun*, *punctuation*. The conversion from Penn Treebank part-of-speech tags set to the reduced POS tag set is straightforward, (e.g., *NN(simple noun)* $\rightarrow$ *noun*, *NNP(proper noun)* $\rightarrow$ *noun*, *VBN(participle verb)* $\rightarrow$ *verb*, etc).

Third, failures occur when there is a dependency between an *appositional* modifier and a *conjunct*. An appositional modifier of an NP is an NP immediately to the right of the first NP. The role of the appositional modifier is to define or modify that NP (e.g. *Carbon monoxide, the gas responsible for ...*). Note that appositions include parenthesized examples, as well as defining abbreviations in one of these structures. We treat conjunctions asymmetrically: the head of the relation is the first conjunct and other conjunctions depend on it via the conj relation.

Given the above three technical difficulties, we restrict our experiments to pairs $\langle T, h\rangle$ with text T containing less than 1000 words or 30 sentences, and hypothesis h containing less than 30 words or two sentences.

The raw texts on climate change were obtained by crawling various debates sites. Hence, replacements of special characters were also required like (i.e the words "quot;" or apos with '). We also noticed that, in summing up certain texts hyperlinks ("http: //" sources) are taken into account and included in the summary, which are inconclusive. Hence, we also removed these http links.

V. Assessing the impact of summarisation on textual entailment

We were interested to measure the impact of summarisation on textual entailment. The research hypothesis was that the processing time will be improved, while the accuracy may decrease with some percent. First, we measured running time and entailment performance on initial pairs $\langle T, h\rangle$. Then we measured the running time and entailment performance on pairs formed by the summarized text and the original hypothesis $\langle T^i_{sum}, h\rangle$. During the experiment we changed the summarisation tools, the percentages of summarisation and the size of the training corpus. The experiments were run on an Intel i7CPU with 2.30GHz, 6GB RAM, 8 cores/64. We made two experiments: one with a small number of training corpus (i.e. 50 pairs) and the other experiment with a larger number (i.e. 1000 pairs). The results are depicted in Tables I and III.

TABLE I: Impact of summarisation on textual entailment in case of small training corpus (50 pairs). The initial text T is replaced by its summary computed by three summarisers C4J, SummApp and CNGL.

Training corpus	*Processing Time*	*Accuracy*	*Recall*	*Precision*	*F1*
$\langle T, h\rangle$	44'30"	0.56	0.53	0.58	0.56
$\langle T^1_{C4J}, h\rangle$	5'24"	0.46	0.38	0.47	0.42
$\langle T^1_{SumApp}, h\rangle$	4'32"	0.52	0.30	0.57	0.40
$\langle T^1_{CNGL}, h\rangle$	7'17"	0.54	0.73	0.54	0.62

The measuring parameters can be defined as follows: accuracy is a ratio of correctly items detected to the total instances; precision is the fraction of relevant instances among the retrieved instances. Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances and the F1 measure is the is the harmonic mean of precision and recall.

In case of small training corpus (50 pairs), the processing time of the original pairs is about 45 minutes (line 1 in Table I). By reducing the original text T to its summary of only one sentence, the processing time drops to 5 minutes in case of C4J summariser, to 4 minutes for SumApp, respectively to 7 minutes for CNGL. That is an average of (5'25"+ 4'32"+ 7'17")/3=5'74". Hence, the processing time is reduced with 87%. For this benefit, we expect a decreasing in accuracy. For this small training corpora, the initial accuracy is 0.56. This small value is not surprising, as 50 pairs for training are not enough for building an accurate language model in the given domain. However, when applying summarisation the accuracy has dropped to the best value of 0.54 in case of the CNGL summariser. Interestingly, the F1 measure increases from 0.56 to 0.62 in case of the same CNGL tool. Note that this behavior did not occur for the C4J and SumApp summarisers. Observe also that the processing time for the summary outputted by CNGL is higher than the other two summarisers. This might indicate the the summary T^1_{CNGL} of CNGL is larger than the summaries T^1_{SumApp} or T^1_{C4J}.

We see two possible conclusions for this experiment. If the user somehow trusts the summariser (CNGL in our case), than the decreasing time drops significantly (from 44'30" to 7'7"), while the accuracy decreases with two percentages. If there is no assessment of the summariser, the average result shows that, after summarisation in one sentence, the entailment decision is close to random guess (i.e, 0.509 accuracy).

As C4J seems to provide more relevant summaries than the other tools for the computing entailment, we performed a further experiment on this summarizer only. We were interested in the entailment performance, when the summary is not reduced to only one sentence, but it can have different percentages from

TABLE II: Applying different summing thresholds for the C4J tool on a small training size (50 pairs).

Summarisation	*Processing time*	*Accuracy*	*Recall*	*Precision*	*F1*
$\langle T, h\rangle$	44'30"	0.56	0.53	0.58	0.56
$\langle T^{50\%}_{C4J}, h\rangle$	35'51"	0.52	0.61	0.53	0.57
$\langle T^{60\%}_{C4J}, h\rangle$	34'32"	0.50	0.53	0.51	0.52
$\langle T^{70\%}_{C4J}, h\rangle$	35'20"	0.54	0.57	0.55	0.56

TABLE III: Impact of summarisation on textual entailment in case of larger training corpus (1000 pairs). The initial text T is replaced by its summary computed by three summarisers C4J, SummApp and CNGL.

Training corpus	*Processing time*	*Accuracy*	*Recall*	*Precision*	*F1*
$\langle T, h\rangle$	16h20'	0.54	0.68	0.54	0.60
$\langle T^1_{C4J}, h\rangle$	2h1'	0.51	0.99	0.51	0.67
$\langle T^1_{SumApp}, h\rangle$	4h40'	0.51	0.93	0.51	0.66
$\langle T^1_{CNGL}, h\rangle$	03h4'	0.52	0.98	0.51	0.67

the initial text. Table II shows the performance metrics when the summary represents 50%, 60%, and 70% from the initial text. The processing time decreases with 20%. The accuracy decreases, but the F1 measure remains the same. However, experiments on larger corpus are required to investigate which would be the best summarisation percentage such that the textual entailment decision to be accurate.

We did the same analyses for a larger corpus of 1000 training pairs. For a corpus of 1000 pairs, the processing time is 16 hours and 20 minutes (line 1 in Table III). After applying summarisation, the required time drops significantly in the range of 2-5 hours, depending on each summariser. Actually, the processing time for pairs $\langle T^i_{sum}, h\rangle$ is at least four times smaller that the initial time of 16 hours and 20 minutes. The initial accuracy of 0.54 is still small, leading the conclusion that 1000 training pairs is still a small number for the textual entailment machinery. Interestingly, the F1 measure increased after summarisation, from 0.6 to 0.66-0.67. And this happened for all three summarisers, as depicted in Fig. 2). One possible interpretation is that the irrelevant text for textual entailment has been removed during summarisation. That is, T^1_{sum} contains sufficient relevant information to infer the hypothesis h. From this perspective, summarisarion acts as a data preprocessing step, in which irrelevant text for inferring h is removed before searching for entailment between T and h.

Fig. 3 shows that the accuracy does not have major fluctuations, decreasing with 0.1 and 0.2 from the initial values. From the point of view of recall, CNGL has the highest value, 0.74 and from the precision part the SummaryApp has the maximum 0.57.

Based on the experiments with 1000 pairs of global warming arguments, Fig. 2 that the CNGL summariser has the best outcome for the accuracy (0.52). In Table III, the recall has maximum value of 0.992 for the Classifier4J, being followed

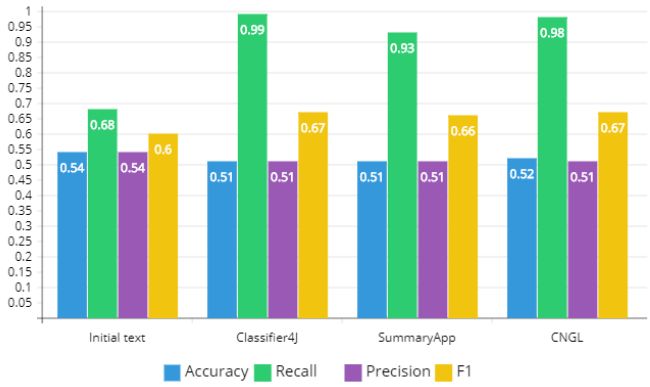

Fig. 2: F1-measure is better in case of summarisation for a corpus of 1000 training pairs$\langle T^1_{sum}, h\rangle$

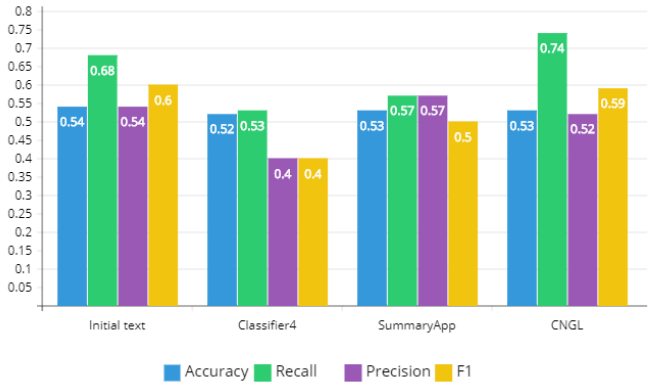

Fig. 3: $\langle T, T^1_{Summarize}\rangle$

by the CNGL with 0.9841. This means that the number of correct entailment proposed are equal with the total number of the entailments that should have been returned. Also, the precision outcome (see Table III) for CNGL is 0.516, that are the instances that have been retrieved over the total amount of relevant instances.

VI. Conclusions

We investigated here how textual entailment can be influenced by text summarization. We run various experiments on pairs of texts and hypotheses from the climate change domain.

We investigated how summarisation can be used to decrease the processing time of textual entailment. The results did indicate a large decrease of processing time, with a small decrease in accuracy.

Two limitations of our experiments are: First, the experiment were limited to 1000 pairs. That number is still small for building a relevant language model, given the search space of textual entailment. We plan to re-run the experiments with larger corpus on a more powerful hardware. Second, we limited here to three summarising tools. We also need to investigate other summarisers and to compare our ranking of summarisers with similar rankings in the literature. Based on

the above two limitations, the experiments are rather preliminary. Still these preliminary results suggest that interleaving textual entailment and summarisation could bring benefits to both domains. On the one hand, summarisation can help to compute entailment decisions more quickly with a quite similar accuracy. On the other hand, textual entailment can be used to automatically assess the quality of the generated summaries.

VII. Acknowledgement

We thank the reviewers for their comments. The dissemination of this paper was supported by the Technical University of Cluj-Napoca through the research grant no. 1996/12.07.2017, Internal Competition CICDI-2017.

References

[1] G. G. Chowdhury, "Natural language processing." *ARIST*, vol. 37, no. 1, pp. 51–89, 2003.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[3] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey." *CoRR*, vol. abs/1503.04069, 2015.

[4] J. Bos and K. Markert, "Combining shallow and deep NLP methods for recognizing textual entailment, proceedings of the first challenge workshop recognising textual entailment," pp. 65–68, 11-13 April 2005.

[5] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL recognising textual entailment challenge," in *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005. [Online]. Available: http://www.cs.biu.ac.il/ glikmao/rte05/

[6] ——, "The PASCAL recognising textual entailment challenge," in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 177–190.

[7] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, "The third PASCAL recognizing textual entailment challenge," in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, ser. RTE '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 1–9.

[8] O. Glickman, I. Dagan, and M. Koppel, "Web Based Probabilistic Textual Entailment," pp. 33–36, 11-13 April 2005.

[9] D. Perez, E. Alfonsecaia, and P. Rodrguez, "Application of the bleu algorithm for recognising textual entailments," in *Proceedings of the Recognising Textual Entailment Pascal Challenge*, 2005.

[10] S. Bayer, J. Burger, L. Ferro, J. Henderson, and A. Yeh, "MITREs submissions to the EU Pascal RTE challenge, proceedings of the first challenge workshop recognising textual entailment," pp. 41–44, 11-13 April 2005.

[11] E. Newman, N. Stokes, J. Dunnion, and J. Carthy, "UCD IIRG Approach to the textual entailment challenge, proceedings of the first challenge workshop recognising textual entailment," pp. 53–56, 11-13 April 2005.

[12] R. Delmonte, S. Tonelli, M. Boniforti, A. Bristot, and E. Pianta, "VENSES - a linguistically-based system for semantic evaluation, proceedings of the first challenge workshop recognising textual entailment," pp. 49–52, 11-13 April 2005.

[13] J. Herrera, A. Penas, and F. Verdejo, "Textual entailemnt recognition based on dependency analysis and wordnet, proceedings of the first challenge workshop recognising textual entailment," pp. 21–24, 2005.

[14] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "Bleu: a method for automatic evaluation of machine translation," 2001.

[15] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[16] S. Pais, G. Dias, K. Wegrzyn-Wolska, R. Mahl, and P. Jouvelot, "Textual entailment by generality," *Procedia - Social and Behavioral Sciences*, vol. 27, pp. 258 – 266, 2011, computational Linguistics and Related Fields. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042811024335

[17] D. Toniuc and A. Groza, "Climebot: An argumentative agent for climate change," in *Intelligent Computer Communication and Processing (ICCP), 2017 13th IEEE International Conference on*. IEEE, 2017, pp. 63–70.

[18] R. Szabo and A. Groza, "Analysing debates on climate change with textual entailment and ontologies," in *Intelligent Computer Communication and Processing (ICCP), 2017 13th IEEE International Conference on*. IEEE, 2017, pp. 39–46.

[19] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, pp. 155–164, April 1958.

[20] H. P. Edmundson, "New methods in automatic extracting," *J. ACM*, vol. 16, no. 2, pp. 264–285, 1969.

[21] K. Sprck Jones, "Automatic summarising: factors and directions," in *Advances in Automatic Text Summarization*, I. Mani and M. T. Maybury, Eds. MIT Press, 1999, ch. 1, pp. 1–12, cmp-lg/9805011.

[22] E. Riloff and J. Shepherd, "A corpus-based approach for building semantic lexicons," in *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, 1997, pp. 109–116.

[23] L. Michelbacher, S. Evert, and H. Schutze, "Asymmetric association measures, proceedings of the international conference on recent advances in natural language processing (ranlp 2007)," 2007.

[24] P.-N. Tan, V. Kumar, and J. Srivastava, "Selecting the right objective measure for association analysis." *Inf. Syst.*, vol. 29, no. 4, pp. 293–313, 2004.

[25] G. Dias and R. Mukelov, "Unsupervised graph-based discovery of general-specific noun relationships from web corpora frequency counts. proceedings of the 12th international conference on natural language learning (conll 2008)," pp. 147–153, 2008.

[26] G. Cleuziou, D. Buscaldi, V. Levorato, and G. Dias, "A pretopological framework for the automatic construction of lexical-semantic structures from texts." in *CIKM*, C. Macdonald, I. Ounis, and I. Ruthven, Eds. ACM, 2011, pp. 2453–2456.

[27] B. Magnini, R. Zanoli, I. Dagan, K. Eichler, G. Neumann, T.-G. Noh, S. Pado, A. Stern, and O. Levy, "The Excitement Open Platform for textual inferences," *ACL 2014*, p. 43, 2014.

[28] D. Ferrucci and A. Lally, "UIMA: an architectural approach to unstructured information processing in the corporate research environment," *Natural Language Engineering*, vol. 10, no. 3-4, pp. 327–348, 2004.

[29] E. Hovy, "Text summarization," in *The Oxford Handbook of Computational Linguistics*, ser. Oxford Handbooks in Linguistics, R. Mitkov, Ed. Oxford: Oxford University Press, 2003, ch. 32, pp. 583–598.

[30] N. Lothian, "Classifier4j," http://classifier4j.sourceforge.net/, 2003.

[31] "Summaryapp," https://thetokenizer.com/2013/04/28/build-your-own-summary-tool/.

[32] S. McQuillan, "CNGL Summarizer," https://github.com/CNGLdlab/CNGLSummarizer, 2012.

[33] L. Kelly, J. Leveling, S. McQuillan, S. Kriewel, L. Goeuriot, and G. Jones, "Report on summarization techniques," *Khresmoi project deliverable D*, vol. 4, no. 4, 2013.

[34] A. Lotan, A. Stern, and I. Dagan, "Truthteller: Annotating predicate truth." in *HLT-NAACL*, 2013, pp. 752–757.