A Proposition-Based Abstractive Summariser

Yimai Fang, Haoyue Zhu, Ewa Muszyńska, Alexander Kuhnle, Simone Teufel
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0FD, United Kingdom
{yf261,hz315,emm68,aok25,sht25}@cam.ac.uk

Abstract

Abstractive summarisation is not yet common amongst today's deployed and research systems. Most existing systems either extract sentences or compress individual sentences. In this paper, we present a summariser that works by a different paradigm. It is a further development of an existing summariser that has an incremental, proposition-based content selection process but lacks a natural language (NL) generator for the final output. Using an NL generator, we can now produce the summary text to directly reflect the selected propositions. Our evaluation compares textual quality of our system to the earlier preliminary output method, and also uses ROUGE to compare to various summarisers that use the traditional method of sentence extraction, followed by compression. Our results suggest that cutting out the middle-man of sentence extraction can lead to better abstractive summaries.

1 Introduction

Abstraction, rather than extraction, is generally seen as the more desirable method of performing automatic summarisation. It implies generating an entirely new text from the information contained in the input text. Amongst its advantages is the promise of better information packaging: conveying more information using less output text. But end-to-end abstractive summarisation is still uncommon in current systems, mostly due to the problems with NLP text analysis and knowledge representation that would plague any "deeper" method. Current research into abstractive summarisation therefore mainly concerns compressing individual sentences or merging sentences of similar content. Sentence compression using machine learning and/or constraint-solving methods is an active area of research. Common methods use syntactic, lexical and discourse-based features to determine which words should be dropped or paraphrased (Knight and Marcu, 2000; McDonald, 2006; Clarke and Lapata, 2008; Cohn and Lapata, 2007, 2008; Yoshikawa et al., 2012). More recently, neural language model has also been applied to the problem (Rush et al., 2015). For multi-document summarisation, sentence fusion (Barzilay and McKeown, 2005) allows the synthesis of common information across documents in one sentence, using grammatical dependencies as a substitute for a semantic representation. These methods have the ability to produce high-quality output, but their sentence generation is a local step in the summarisation pipeline, i.e. isolated from the content selection, which is essentially global. The only exceptions we know are Martins and Smith's (2009) system, and Nishikawa's (2014) system for Japanese text, both of which optimise sentence selection jointly with sentence compression.

A different paradigm for abstractive summarisation would be to avoid the "middle man" of sentence extraction altogether, and to operate over propositions instead. Such a system would first transform the text into sub-sentential semantic units, then perform summarisation operations over these (possibly performing inference and creating new semantic units), and finally use NL generation techniques to verbalise the semantic units that make up the final summary. Kintsch and van Dijk (1978, henceforth KvD) present one such summarisation technique. They use a tree of connected propositions to simulate the human memory. While the text is processed incrementally, new propositions are attached to the tree

by argument overlap, and old propositions are pruned according to simulated memory limitations. The final summary is based on the best-connected propositions. We have presented in previous work (Fang and Teufel, 2016) an implementation of KvD's core ideas. However, it did not have the ability to generate new text from the propositions, and we were thus forced to use sentence extraction for the creation of a compromise summary output. We were nevertheless able to show that the content selection of our summariser outperforms state-of-the-art extractive summarisers.

The contribution in the current paper is the combination of our summariser with an existing NL generator that can verbalise the summary propositions. The propositions we use are created by aggregating grammatical dependencies gained from a syntactic parse with the Stanford Parser (Klein and Manning, 2003), in an attempt to create more meaningful semantic units similar to KvD's original concept. This close-to-surface representation allowed us ease of operation and robustness, but the grammatical relations do not contain enough semantic information to regenerate from them. During our research, we decided to experiment with the ACE processor[1], allowing parsing and generation from Minimal Recursion Semantics (MRS) representations (Copestake et al., 2005) and related formalisms such as Dependency Minimal Recursion Semantics (DMRS) (Copestake, 2009). For sentences containing summary propositions, our generation module aligns Stanford grammatical dependencies with their corresponding DMRS components (called Elementary Predicates or EPs). New text is then generated by selecting the minimal set of EPs such that it fulfils two conditions: 1. it covers all the information in the desired proposition from our summariser and 2. it contains all the EPs necessary for successful generation with ACE. The second condition ensures grammaticality.

How should one assess the quality of a summary sentence that minimally expresses the information in a proposition? Our first evaluation aims at measuring the grammaticality and truth preservation of the generated sentences using a human experiment. It is clear that producing grammatical, semantically and pragmatically interpretable text is a necessary goal for any abstractive technology. Truth-preservation is a property that also needs to be evaluated carefully whenever sentential material is manipulated. It is easy to accidentally introduce effects that would distort the meaning of the new sentence, be it by inappropriate referring expressions, by dropping restrictive relative clauses and thus changing the truth-conditional conditions of the sentence, or by many other subtle unintentional changes. In our evaluation, we ask humans to interpret the shortened sentence together with its original context in order to detect possible truth distortions.

Our second evaluation uses the ROUGE metric (Lin, 2004) to evaluate the overall content selection of our abstractive end-to-end method. We cannot directly compare our output to that of an isolated sentence compressor; we first need to select the sentences. We therefore build a pipeline system that uses our summariser as the sentence extractor, and then compresses the summary sentences with two competitive sentence compression methods (Clarke and Lapata, 2008; Cohn and Lapata, 2007). This gives the extraction–compression route a fair chance because our system as a sentence extractor was found to be the best system evaluated in our previous work (Fang and Teufel, 2016). As a further comparison system, we additionally use the next best "generation" algorithm that can express the information in a proposition: one that simply extracts all words that gave rise to the proposition (cf. section 4.2). This is the output mechanism we used in (Fang and Teufel, 2014).

Our results show that our system produces sentences of a good textual quality, and that the overall content selection (as measured in ROUGE) rivals current sentence-selection, while achieving much stronger compression than traditional sentence compressors do. We see two possible reasons why leaving out the middle-man of sentence selection is advantageous in abstractive summarisation. General sentence compression systems are trained to recognise material that tends to be dropped in a "general case". But without information about global discourse effects in the overall text, such a compressor cannot choose particular information selectively inside the sentence. The sentence extractor, which is run independently of it, would typically have selected sentences based on different criteria. In contrast, our system has access to global information about what connects the best propositions, which, as we would argue, enables it to perform better content selection. Secondly, the fact that our system works on propositions, i.e.,

[1]The Answer Constraint Engine, `http://sweaglesw.org/linguistics/ace/`

smaller semantic units, also affords it better information packaging abilities. This is shown by the fact that our abstractive summariser compresses sentences more heavily than traditional systems, because it uses an input sentence only as "raw material" to realise the small piece of information which it has determined to be essential.

In what follows, we will review previous work including the operation of the underlying proposition-based summariser and the NL generator we use (section 2). We will then explain our generation algorithm (section 3), describe our evaluations (section 4), and conclude with a discussion of our results.

2 Background

We now introduce the two modules we use: the summariser that chooses the summary propositions, and the grammar on which the generation operates.

2.1 Proposition-based summarisation

Our proposition-based summariser was introduced in Fang and Teufel (2014) and improved in Fang and Teufel (2016). The summariser maintains a coherence tree, which simulates human working memory. It processes the text sentence by sentence, incrementally adds new propositions to the tree, and prunes old propositions off the tree. Propositions are attached to existing propositions in the tree with which they share arguments (semantic participants). They are "forgotten" if the summariser predicts that no new propositions are likely to be attached to them, using a "leading-edge strategy" that prefers keeping recent nodes that are high-level (i.e. of small depth) on the tree. The idea of applying a graph algorithm on a semantic representation is not new to unsupervised summarisation (Vanderwende et al., 2004), but we have shown that the incremental processing of our model is superior to computing centrality on a single graph that represents the full text (Fang and Teufel, 2016).

The summariser is inspired by KvD's theory of text comprehension. The computational model we present is a simplification of the theory in two ways: 1. It does not generalise propositions or create meta-statements (macro-propositions) for specific domains. (These operations could in principle be performed if NLP technologies such as robust inference and entailment recognition were in place.) Instead it defines the summary-worthiness of a proposition in the simplest possible way, as the number of cycles during which it is retained in memory. 2. Rather than using "concepts" as arguments in the propositions (which would require human intelligence), we create propositions based only on a syntactic parse, and only containing textual tokens, not intelligent "concepts". Our propositions are comparable to the subject-verb-object triples in Genest and Lapalme (2011), but we allow more kinds of predication such as adjectival and prepositional modifications to be propositions in their own right, which means these pieces of information can be selected independently. For example, "*The discovery of the element revolutionised fire-lighting*" gives rise to the propositions *revolutionised (the discovery, fire-lighting)* and *of (the discovery, the element)*. We use coreference resolution and models of semantic relatedness (such as lexical chains and cosine similarity in a vector space) to determine the overlap between these arguments.

Starting from a list of summary propositions found to be important by the summariser, how should we generate a textual summary from it? Doing so is not only important for the usability of the summariser, but also for automated evaluation – ROUGE evaluates the quality of summaries by comparing them to human-written gold standard summaries. It cannot evaluate abstract propositions without surface forms. To solve this problem, we previously made our summariser extract full sentences that contain the summary propositions. However, this does not fully demonstrate the fine granularity of the content selection of our proposition-based summariser, and the advantages this granularity brings. We also ideally want grammatical sentences that *only* realise the information from the propositions selected by the summariser. We do not want the summary to be diluted by any other content that happened to co-occur with it in the same input sentence. In this paper we present a solution, namely NL generation from propositions.

2.2 ACE and The English Resource Grammar

ACE is one of the processors designed to work with DELPH-IN HPSG wide-coverage, linguistically motivated grammars such as the English Resource Grammar (ERG) (Flickinger, 2000; Flickinger et al.,

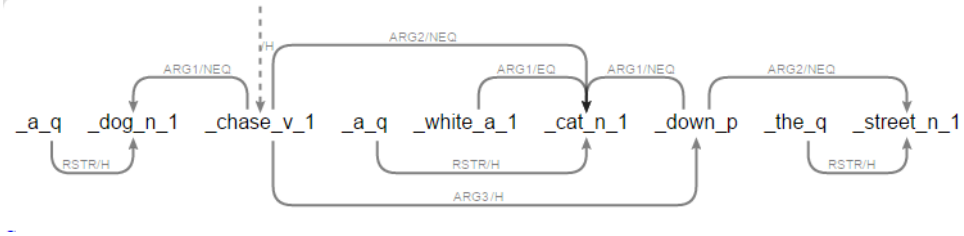

Figure 1: A DMRS graph for the sentence "*A dog chased a white cat down the street.*"

2014), a broad-coverage, symbolic, bidirectional grammar of English. It was developed as part of the DELPH-IN initiative[2] and LinGO[3] project. The ERG uses Minimal Recursion Semantics (MRS) (Copestake et al., 2005) as its semantic representation. In contrast to this, the kind of generation that is traditionally used in summarisation to realise proposition-like units is much shallower, e.g. SimpleNLG as used by Genest and Lapalme (2011, 2013). The MRS format can be transformed into a more readable Dependency Minimal Recursion Semantics (DMRS) graph (Copestake, 2009), which represents its dependency structure. An example of a DMRS graph is shown in Figure 1. The nodes correspond to predicates (EPs); edges (links), represent relations between them.

DMRS graphs can be manipulated using two existing Python libraries. The `pyDelphin` library[4] is a more general MRS-dedicated library. It allows conversions between MRS and DMRS representations but internally performs operations on MRS objects. The `pydmrs` library[5] (Copestake et al., 2016) is dedicated solely to DMRS manipulations.

3 NL generation from propositions

To generate, a deep semantic representation for generation is needed – as well as the propositions, which represent the content we want to include in the summary. But for historic reasons, the front-end of our summariser is the Stanford parser. At the time, propositions based on dependency relations seemed to provide the best fit to KvD's notion of propositions. After realising the shortcomings of shallow generation by extraction, and after the ACE generator became available, we started experimenting with deep generation of summaries using DMRS, as described in this paper. As a downside of this development, we now have to run two parsers on each sentence that is involved in the final summary.

We iteratively generate summary text for each proposition in the ranked list of propositions output by our summariser described in section 2.1, until the summary length limit is reached. For each proposition to be verbalised, we use the original sentence that contains it as the source, parse it using ACE, and then manipulate its DMRS representation so as to generate a (much shortened and) grammatical summary sentence, which should then express the meaning of the proposition in a human-digestible form.

Because propositions from a sentence are ranked independently, as we go down the ranked list, it is possible that we encounter a proposition that shares the source sentence with one or more propositions that were verbalised a few iterations ago. In this case, we restart the generation on that source sentence, and require the new formulation to include the information of both the new and the old propositions. The previously generated summary sentence is then replaced by the new formulation.

3.1 Initial nodes selection

The first step of generating a sentence is to select the initial set of nodes (EPs) in the DMRS representation of the source sentence. This initial set corresponds to the one or more propositions we want to verbalise from the source sentence. A proposition contains information of the index of the source sentence, as well as the textual tokens of its functor and arguments. The character offset of a token is useful because a token in the dependency parse (and therefore in the proposition) may not necessarily align with an EP of

[2]Deep Linguistic Processing with HPSG, `www.delph-in.net`
[3]Linguistic Grammars Online, `lingo.stanford.edu`
[4]`https://github.com/delph-in/pydelphin`
[5]`https://github.com/delph-in/pydmrs`

ERG. For example, the Stanford parser analyses "*didn't*" as two tokens `did` and `n't`, while ERG treats it as one grammar predicate called `neg`. Referring to the character span of `neg`, we are able to select it if the propositions contain either token.

We also need a strategy for unknown words, as ACE does not allow generation from a DMRS if any of its EPs contains an unknown word. We handle this by a temporary replacement mechanism where dummy words stand in for unknown words in the input text.

3.2 Node set expansion for grammaticality

After obtaining the initial node set, our algorithm appends additional nodes and removes existing nodes, based on syntactic information from the ERG parse. The most essential general rule is to always add EPs which are syntactic arguments of already selected EPs. These arguments are indicated by NEQ and HEQ links in the ERG. The ERG has more sophisticated mechanisms than shallower parsers for dealing with subcategorisation in general, which should improve the grammaticality of our results. Our rules include adding compulsory prepositional complements of verbs, adding quantifiers, and dismantling sentence coordination.

We have devised specialised rules for cases when the syntactic parse is more superficial than the ERG's analysis. For instance, the ERG can handle complicated cases of quantification, e.g. "*most of the students*", by using additional meaning-bearing EPs which express semantic relationships such as a part-of relationship or a quantity relationship. In the Stanford dependencies, however, the quantifier "*most*" is the head of the noun phrase, modified by a prepositional phrase "*of the students*". We therefore prevent the undesirable outcome where "*most*" is selected on its own for the summary, by including the semantic head "*the students*", which is connected to the predicate `part_of` via an NEQ link.

As an example, consider a source sentence encountered in our evaluation:

> *In Britain, for example, the dull weather of winter drastically cuts down the amount of sunlight that is experienced which strongly affects some people.*

Extracting the textual tokens that gave rise to the selected summary propositions produces an uninformative and ungrammatical sentence for the summary:

> *In Britain, for example, the weather cuts down the amount strongly affects.*

But our ERG-based generation preserves the semantic head of the quantity relationship, and the obligatory object of the transitive verb:[6]

> *In Britain, the weather cuts down the amount of sunlight, which affects some people, strongly.*

We have also written rules for abstract DMRS predicates such as compound, implicit conjunction, and apposition. We use about ten specialised rules to treat certain constructions, such as "*in order to*", and phrasal verbs such as "*keep from*". The system could be strengthened in the future by the systematic addition of more such rules.

3.3 Node set expansion for graph connectivity

Generation with ACE requires a connected DMRS graph. In this step we ensure connectivity of the subgraph using an undirected graph algorithm. But the subgraph consisting of the EPs selected so far (henceforth "the subgraph"), which we will use for generation, may or may not be connected: If the initial set of EPs was already connected, the expanded set created by the algorithm in section 3.2 will certainly remain connected. If the initial set of EPs was not connected, however, the algorithm for grammaticality can sometimes make the set connected, but there is no guarantee.

Making a disconnected subgraph connected by including the minimum set of additional nodes corresponds to the NP-hard Steiner tree problem (Hwang et al., 1992). Our greedy approach to this problem is to iteratively grow the largest connected component. An iteration starts with the largest connected

[6] Please note that the adverbial modifier "*strongly*" appears in a different place from the source sentence, but is still grammatical. This is an effect of the generator's degree of freedom when verbalising a semantic representation, discussed in section 3.5.

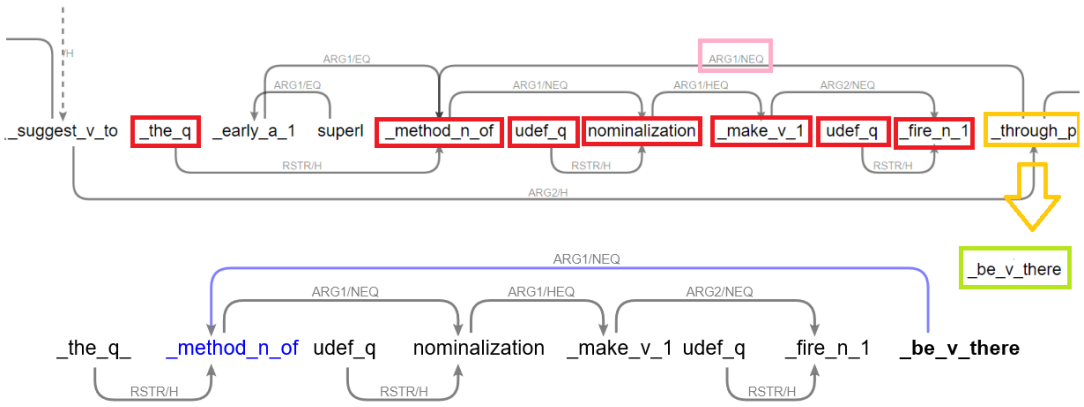

Figure 2: A "*there be*" generation.

component in the subgraph. We want to include one additional node into the subgraph in each iteration, and we select such a node among the non-selected nodes that are adjacent to this connected component. First, we prefer including a node if that node can join the maximum number of other connected components to this component. Second, in case of a tie, we prefer a node that is positioned in the EP sequence (i.e. in textual order) between this connected component and an other connected component. This serves as a heuristic that guides us towards the nodes that are adjacent to the other components. Third, as an additional tiebreaker, we prefer a node that is connected to this connected component via an NEQ link over one connected via an EQ link. After adding the additional node to the subgraph, the next iteration starts with the currently largest connected component in the subgraph. The process continues until there is only one connected component in the subgraph, i.e. the subgraph is connected.

3.4 Node modification

We always want to generate full, grammatical sentences. For propositions that are expressed as noun phrases in the original text, the best solution would be to reformulate the phrase in a verbal predicate. But this is hard, so instead we devised a graph-based procedure that generates "*there be*" sentences for them. This algorithm works by finding the local top node for the set of selected nodes (called `s_n`), and changing this local top node to `_be_v_there_`.

Figure 2 shows a part of the subgraph of the DMRS representation of the sentence

> *Studies of primitive societies suggest that the earliest method of making fire was through friction.*

After expanding the node set from the initial set indicated by our selected proposition, we arrive at the selected node set as shown in the red boxes, namely {`_the_q`, `_method_n_of`, `udef_q`, `nominalization`, `_make_v_1`, `udef_q`, `_fire_n_1`}. The local top node for the node set is `_through_p` (shown in the yellow box); it is connected to the noun predicate `_method_n_of` with an outgoing NEQ link (shown in the pink box), and it is the only link connected with the set. Therefore, we replace the local top with the predicate `_be_v_there_`, and pass the resulting DMRS subgraph to the generator, resulting in the summary sentence

> *There is the method of making fire.*

While this sentence is not perfectly natural, it verbalises the information about fire-making methods reasonably fluently, without introducing unnecessary information from the source sentence that was not selected for the summary.

3.5 Selection among generations

The final step of our method concerns the selection of the best surface sentence generated by ACE. Given a DMRS, ACE generates a list of sentence variants covering the DMRS's semantics, which can

differ by active vs. passive tense, verb alternations, order variations and various other effects. ACE ranks the variants using a statistical model according to naturalness. If a sentence is parsed into EPs, the regenerated sentence from these EPs may therefore be different from the original (e.g. "*Kim gave a book to Sandy*" can turn into "*Kim gave Sandy a book*"). But sentence reformulation is undesirable for us, because we want the shortest sentence that is also closest in meaning and external form to the original sentence – and not necessarily the "most likely" sentence. We therefore select, from the top five generated sentences, the one that shares the longest common subsequence (LCS) with the original sentence.[7]

Of course, Stanford and ACE do not always give the same sentence an equivalent analysis, and not all ACE parses and generations succeed. ACE parsing could fail for different reasons, mostly in our case due to very complex and long sentences, unforeseen syntactic constructions and unusual punctuation. We used our corpus of 108 documents described below to measure its coverage, which is 86.5%.[8] Our backoff strategy in case of parsing or generation failure is to produce the sequence of all tokens which are involved in the proposition, as we did in Fang and Teufel (2014). This primitive generation technique, also a baseline system we use in the following section, is quite likely to produce ungrammatical output, but it is generally applicable to any proposition.

4 Evaluation

For both evaluations we perform, we use the IELTS summary corpus, which we introduced in Fang and Teufel (2016). The IELTS is a standardised test of English proficiency for non-native speakers. The texts we use are taken from the academic reading sections of the official practice tests. We use all 108 such documents available in *Cambridge IELTS 1–9*,[9] of which we randomly sampled 31. For each of the 31 texts, we commissioned four 100-word summaries by 15 native, near-native, or highly proficient speakers of English. We chose this corpus over the commonly-used news corpora for evaluation, because the IELTS texts are not in the journalistic genre, where the lead sentences are usually abstract-like, yet they are naturally occurring, generally understandable and of high editorial standard. They have the additional advantage that they do not need specialised world knowledge in the areas of politics or economics, as news articles (and particularly the frequently used Wall Street Journal) often do. This is beneficial if one wants to study text understanding-based forms of summarisation.

We use human judgement to evaluate the textual quality of individual generated sentences, and use the automatic ROUGE metric to evaluate the content selection ability on entire summaries.

4.1 Systems

We test four kinds of generation mechanisms for summarisation here (Figure 3): 1. OurAbs, our KvD-based summariser with NL generation, which realises the summary propositions directly as new text; 2. various sentence extraction systems (without sentence compression); 3. various combinations of sentence extraction systems followed by sentence compression systems; and 4. OurTok, our KvD summariser with the primitive generation technique, which only outputs the word tokens involved in summary propositions in textual order, without any attempt at generating fully grammatical sentences.

Both OurAbs (category 1) and OurTok (category 4) are based on first deriving a ranked list of summary propositions, where the importance (weight) of a proposition is determined by the KvD-style simulation of text understanding (cf. section 2.1). In principle, a summariser in categories 2 and 3 could also use the same information to extract sentences (followed by a compression stage for category 3). For instance, an extractor can choose sentences such that the sum of the weights of the propositions from these sentences is maximised. We have in fact implemented such a sentence extractor and formally

[7] If there is a tie, we choose the sentence with fewer words. If candidates of the maximum LCS are equally long, we choose the top-ranked generation.

[8] We also implemented a simple sentence simplification method that splits unparsed sentences at top-level S coordination nodes, which increased coverage to 90%. But we do not further report on this method here because its ROUGE results suffered slightly.

[9] For example, *Cambridge IELTS 9: authentic examination papers from Cambridge ESOL*, Cambridge University Press, ISBN: 9781107615502.

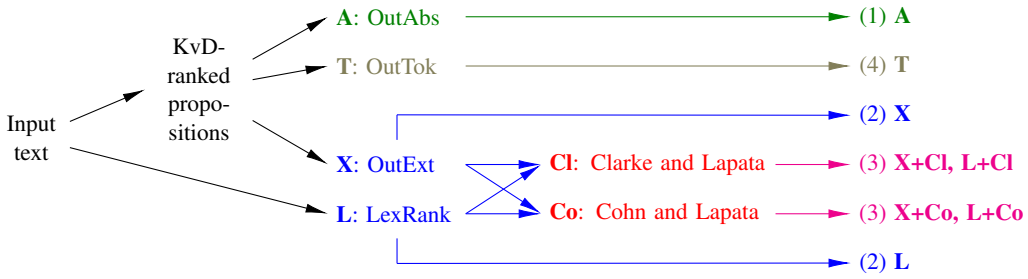

Figure 3: The 8 summarisation systems in 4 categories.

	Grammatical			Truthful		
	OurAbs	OurTok	Tie	OurAbs	OurTok	Tie
Average	20	5	15	25.3	5.3	9.3
Total	120	30	90	152	32	56

Table 1: Text quality evaluation: number of "higher" judgements given.

evaluated its performance in Fang and Teufel (2016); we call it OurExt here.

Alternatively, the sentence extraction module could be a shallower summariser that is not based on propositions. For instance, LexRank (Erkan and Radev, 2004) is a summariser that internally represents the input document as a graph of sentences. The inter-sentential similarity dictates the edges and their weights, and the centrality algorithm computes the value of a sentence by the probability of a random walk arriving at it. It does not build the graph incrementally, and it only measures the importance of sentences but not meaning units.

We use two sentence compression methods in combination with OurExt and LexRank. Clarke and Lapata (2008) use integer linear programming (ILP) to find the optimal compression per sentence within linguistic constraints. We choose its unsupervised version,[10] which requires a language model. We trained a trigram language model on the 100M-word British National Corpus (BNC) with interpolated Kneser-Ney smoothing and the "unknown word" token. Cohn and Lapata (2007) present a supervised tree-to-tree transduction method for sentence compression.[11] It is a variation of the Noisy Channel model for sentence compression first introduced by Knight and Marcu (2000). Instead of the Broadcast News Corpus they used, we trained the tree rewriting model on the full Written News Corpus, which only became available later (Clarke and Lapata, 2008). Following them, we aligned the words between the original and the compressed sentences using GIZA++ (Och et al., 1999), with additional pairs added where each word in the lexicon is aligned with itself, and symmetrised the alignment using the intersection heuristic (Koehn et al., 2003). We used the Stanford PCFG Parser (Klein and Manning, 2003) to obtain the sentence parses, and the above-mentioned BNC language model for training and decoding.

Both compressors were run on all input sentences.[12] The sentence extractor (OurExt or LexRank) computes the summary-worthiness of the uncompressed sentences, and then the corresponding compressed sentences are output.

4.2 Textual quality evaluation

We performed a human experiment to evaluate the textual quality of the sentences generated by our new system OurAbs, in comparison to OurTok. This evaluation is performed on individual sentences. We only tested sentences where OurAbs's generation succeeded, as OurAbs's fall-back strategy is identical to OurTok's normal operation.

We recruited six participants for this study. They were provided with 40 sentence triples, each consisting of a sentence from the input document, and both outputs by OurTok and OurAbs based on the same

[10] We used the implementation from `https://github.com/cnap/sentence-compression`

[11] We obtained the implementation from the authors.

[12] The Cohn and Lapata compressor failed to process 8 sentences. If any of them is chosen for summary, the uncompressed sentence is used instead.

	R-1	R-2	R-L	R-SU4
Abstract generation from propositions				
OurAbs (A)	0.364	0.088	0.340	0.131
Sentence extraction with compression				
X + Cl	0.361	0.090	0.335	0.132
X + Co	0.340	0.074	0.321	0.113
L + Cl	0.356	0.077	0.325	0.126
L + Co	0.336	0.067	0.314	0.110
Sentence extraction				
OurExt (X)	0.376	0.122	0.345	0.154
LexRank (L)	0.349	0.087	0.316	0.129
Token extraction for propositions				
OurTok (T)	0.356	0.088	0.336	0.130

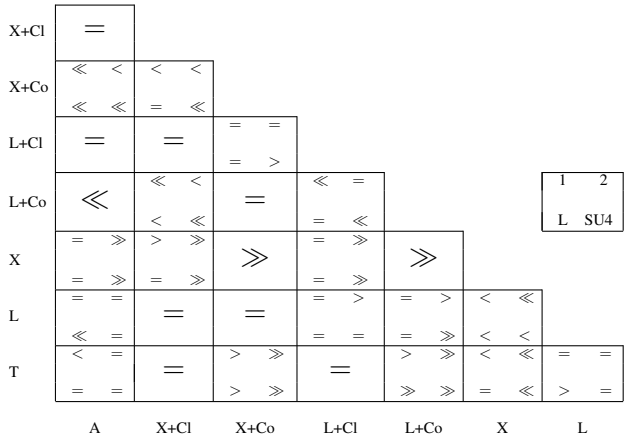

Table 2: ROUGE F-scores and statistical significance of the differences. The four positions in the significance table correspond to ROUGE-1, 2, L and SU4, respectively. "$\gg$" means row statistically outperforms column at $p < 0.01$ significance level; "$>$" at $p < 0.05$ significance level, and "$=$" means no statistical difference detected.

underlying summary proposition(s), in randomised order across triples. They were asked to rank the two summaries for two properties: 1. Which sentence is more grammatical? 2. Which sentence is more truthful to the meaning of the original sentence? They were allowed to judge two summaries as equally good (i.e. tie). The total number of judgements in this evaluation is 240 (40×6) for each question.

The results are given in Table 1. In 50% of cases (120/240), OurAbs was judged as strictly more grammatical than OurTok; as opposed to 12% of cases where OurTok was judged as strictly more grammatical. In terms of truth preservation, OurAbs was judged as distorting the truth less in 63% of cases (152 out of 240), whereas OurTok was seen as more truth-preserving in 13% of cases (32 out of 240). All differences are statistically significant as determined with the sign test at $p = 0.05$. Seeing that the new system improved or did not decrease grammaticality in 87% of cases where generation was successful, and improved or did not decrease truth-preservation in 86% of cases, we believe that the new generation technique has increased text quality overall considerably over our previous primitive method, while achieving a very strong compression of 33%.

4.3 Content selection evaluation

We evaluate the quality of content selection using four ROUGE metrics (ROUGE-1, 2, L, and SU4) on the IELTS corpus, and compute the statistical significance using the paired Wilcoxon test (Table 2). Although OurExt is the best system in all metrics, OurAbs performs comparably to it on ROUGE-1 and ROUGE-L. It performs at least as well as the four pipeline systems. This is an achievement because the goal of OurAbs is to generate only the summary propositions, and is therefore only directly comparable to OurTok. It is an improvement over OurTok, which is significantly worse than OurExt in all metrics except ROUGE-L.

To a limited degree, we can compare OurAbs to the pipeline systems that combine sentence extraction with compression, as long as we bear in mind that OurAbs faces a more challenging generation task: it is required to capture specific information with a sentence, instead of what is generally important in that sentence. It performs this task very well, which is shown by a very low compression rate (length of generation divided by length of original, in words) of 33%, and easily beats either sentence extractor combined with Cohn and Lapata's compressor, which compresses to 43%. Under ROUGE, when the compression rate decreases, the shorter sentences give rise to much fewer n-grams, so that the generation imperfections become disproportionately penalised by ROUGE's numerical scores. We see the extreme case in OurExt, which has the highest ROUGE scores but does not compress at all. Even under these adverse conditions, OurAbs still performs indistinguishably from any sentence extractor with Clarke and Lapata's compressor, which, at a compression rate of 67%, benefits under ROUGE from being closer to full-sentence extractions.

5 Conclusions

What we have presented here are the first steps in abstractive summarisation via propositions. What we hope we have shown is that there is promise in using propositions as a functional semantic representation, which works for both content selection and verbalisation of the output. Relying only on text coherence for the memory operations, this content selection process has the flexibility to work across domains.

Our new KvD-based abstractive summariser uses propositions not only for content selection, but also for generation. For this we use a deep generator based on the DMRS formalism. Others have used proposition-like units for selection and generation, but they use a much shallower NL generator and they limit the units either to certain syntactic constructions (Genest and Lapalme, 2011) or to certain domains (Genest and Lapalme, 2013). The addition of the NL generator is of course an important step forward for the usability of our summariser, but it is also an improvement for evaluation, because it allows us to measure the effect of granularity in content selection, without being tied to sentence extraction as our generation method, as we were before. In the future, we also plan to concentrate our efforts on developing a fully automatic proposition-based evaluation.

Our current system produces summary output of a higher quality than our previous primitive output method. For a more extensive comparison, we have constructed the combinations of sentence extractors (both ours and LexRank) with two well-cited sentence compressors. Sentence compression presupposes a prior step of sentence extraction; it is the only other way how abstractive summaries can be constructed. However, this previous step is rarely talked about in the literature. In this work, we have implemented four versions of this extraction–compression model, and found that our system performs at least as well as these systems. But its biggest advantage is that it has better information packaging abilities, as demonstrated by the lower compression rate. This is arguably an effect of our use of propositions.

There are also many ways how the generation component could be made more robust. Despite ERG's high quality, parsing and generation sometimes still produces ungrammatical or awkward text, or fails altogether, in which case our system has to fall back to the crude token-based output method. The main reasons for this is that our system is still not very deep, as there are many cases when neither the KvD step nor the generation step has enough information to select the correct information.

We see the fact that our summariser is not based on sentence extraction for content selection as one of its advantages. However, the generation step itself currently has the limitation that the textual output representing a proposition can only come from a single sentence. This leads to an abundance of bullet point-like short sentences in our automatic summaries, which also makes our abstractive summaries disadvantaged in the ROUGE-based evaluation. Removing this limitation would allow a truly flexible summariser. This could be achieved by knowing the identity of concepts in a document, and aggregating attributes of an entity (i.e. propositions about an entity). As a subproblem of this, we would need to generate appropriate referring expressions, or create discourse-new elements as needed.

The KvD summariser itself is also constantly being developed further. For instance, keeping a distinction between identity and bridging links between arguments is important to the above-mentioned task. On top of this, the recognition of identical propositions and generalised propositions that already exist in text can also be achieved, which will be an initial step towards inference.

References

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *EMNLP-CoNLL*, pages 73–82.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyńska. 2016. Resources for building applications with Dependency Minimal Recursion Semantics. In *Proceedings of the Tenth Language Resources and Evaluation Conference (LREC '16)*.

Ann Copestake. 2009. Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece, March. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Yimai Fang and Simone Teufel. 2014. A summariser based on human memory limitations and lexical competition. In *EACL*, pages 732–741.

Yimai Fang and Simone Teufel. 2016. Improving argument overlap for proposition-based summarisation. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 479.

Dan Flickinger, Emily M. Bender, and Stephan Oepen. 2014. Towards an encyclopedia of compositional semantics. Documenting the interface of the English Resource Grammar. In *Proceedings of the Ninth Language Resources and Evaluation Conference (LREC '14)*, pages 875–881, Reykjavik, Iceland.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2013. Absum: a knowledge-based abstractive summarizer. *Génération de résumés par abstraction*, 25.

Frank K Hwang, Dana S Richards, and Pawel Winter. 1992. *The Steiner tree problem*, volume 53. Elsevier.

Walter Kintsch and Teun A. van Dijk. 1978. Toward a model of text comprehension and production. *Psychological review*, 85(5):363–394.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization – step one: Sentence compression. *AAAI/IAAI*, 2000:703–710.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

André FT Martins and Noah A Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9. Association for Computational Linguistics.

Ryan T McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *EACL*.

Hitoshi Nishikawa, Kazuho Arita, Katsumi Tanaka, Tsutomu Hirao, Toshiro Makino, and Yoshihiro Matsuo. 2014. Learning to generate coherent summary with discriminative hidden semi-markov model. In *COLING*, pages 1648–1659.

Franz Josef Och, Christoph Tillmann, Hermann Ney, et al. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Lucy Vanderwende, Michele Banko, and Arul Menezes. 2004. Event-centric summary generation. *Working notes of DUC*, pages 127–132.

Katsumasa Yoshikawa, Tsutomu Hirao, Ryu Iida, and Manabu Okumura. 2012. Sentence compression with semantic role constraints. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 349–353, Stroudsburg, PA, USA. Association for Computational Linguistics.