Improving and Evaluating Methods for Automatic Factual Question Generation

Alexander Gamble

Hughes Hall

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: ag928@cam.ac.uk

June 16, 2017

Declaration

I, Alexander Gamble of Hughes Hall, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 12853

Signed:

Date:

Abstract

Answering questions about a passage of text can be a useful method of measuring a language learners progress in reading comprehension. For a teacher or examiner, manually creating questions about text for their students can be a time consuming process. More recently, there have been many attempts at creating systems to perform automatic question answering, mostly using 'deep-learning' methods. These system frequently rely on large scale question-answering datasets which have been created through significant manual effort and financial expense. Providing a method to remove the manual process could remove this expenditure of time and money in both of these scenarios.

This project focuses on improving and evaluating methods for automatic question generation. I provide a literature review into different approaches to question generation and the techniques used to evaluate.

I extend an existing system for question generation to include a component for automatically identifying salient portions of text, and a component for 'learning' the type of question to generate. I also provide a re-implementation of a recent neural network approach to automatic question generation. The models are evaluated for lexical and semantic similarity to gold-standard questions. I also evaluate generated questions for overall 'fluency'.

The results suggest that there is not a significant improvement to lexical or semantic similarity of generated questions to gold-standard questions, against the same system without modified components. I provide discussion of these results, and suggestions as to reasons for this performance.

Word Count: 12853

Acknowledgements

Thank you to my supervisor, Dr. Ekaterina Kochmar, for her assistance and guidance during this project.

Contents

Chapter 1

Introduction

1.1 Motivation

Answering comprehension style questions about a passage of text can be a useful method of evaluating how well a reader has understood its salient ideas. For a native speaker of English, questions asked about a given passage can require analytical reasoning or subjective thought to answer. These types of questions could ask for implicit details of the text, or ask for multiple separate details across the text to be aggregated. They are difficult to generate and answer because of these complications.

For a non-native speaker learning English at a basic level, analytical style questions are likely to be too challenging to answer. Instead, questions which evaluate a language learner's understanding of the question itself are more suitable. For these types of question, answers should be extractive of the original source text.

In this project I define these types of questions as *factual*. Factual questions begin with one of the *wh-* words: *who*, *what*, *when*, *where*, *how many*, *whose* and the answer phrase to a given factual question is extractive of the original source text. Factual question generation is the task of creating factual questions from a given passage of text.

Another application for automatically generating factual questions is for the automatic generation of datasets for question-answering systems. Normally these types of datasets are created as part of an expensive and time consuming manual process. Human annotators are required to read text, identify salient portions and reformulate the input sentences into questions. An automatic system to generate questions would alleviate this process and allow for much larger datasets to be created for this task. For example the SQuAD dataset (Rajpurkar et al., 2016) is a collection of over 500 Wikipedia articles with over 100,000 question-answer pairs. As far as I am aware, this is the largest question-answer dataset currently available. This dataset was produced using a Mechanical Turk system, where workers were asked to spend 4 minutes reading and generating questions for each paragraph. From the paper the total monetary cost of paying annotators was over $50,000 and over 6,000 man hours were spent (calculated from the stated worker per hour cost of $9, per document time of 4 minutes, with 100,000 documents (Rajpurkar et al., 2016)).

1.2 Project Structure

This project conducts a review of the existing literature into methods for automatic factual question generation and the techniques for evaluating the generated questions. I also conduct a literature review into techniques for identifying salient sections of text, using extractive and abstractive summarisation methods.

My approach focuses on improving the pipeline of question generation in the system used in Heilman (2011). I extend an existing Java program to include components for performing automatic summarisation and a component which learns the type of *wh-* question to generate.

I run several models based on combinations of these new components on the SQuAD dataset (Rajpurkar et al., 2016) and evaluate the generated questions on their lexical similarity and semantic similarity to gold standard questions.

I also measure the question ‘fluency’ of each of these models - “to what extent is the question stated in grammatical English?” (Yuan et al., 2017). Finally, I also include a re-implementation of a recent end-to-end neural question generation model (Du et al., 2017) and evaluate its quality along the same metrics.

Chapter 2

Related Work

2.1 Rule-based Question Generation Systems

2.1.1 Template Systems

A common approach to automatic question generation is substitution of named entities or noun phrases into pre-generated question templates.

Cai et al. (2006) present NLGML, a markup language designed for automatic natural language generation. This markup language is based on XML and allows for declarative pattern matching and template evaluation based on input text. Lindberg (2013) demonstrates how this markup language can be used to generate questions using prdefined templates. This template system operates over phrase structure trees of some input text. The system allows the user to declare templates which match some parser output, extract portions of structurally matching phrases, and 'fill-in' the blanks of given question templates with these contents.

```
<category>
   <pattern>
     <S>
       <NP person="true">_person_</NP>
```

```
          <VP> <VBD>went</VBD>
            <PP> <TO>to</TO> <NP location="true">
               _place_</NP> </PP>
          </VP>
<star /> </S>
      </pattern>
      <template> Where did _person_ go? </template>
      <template> Who went to _place_? </template>
      <template> Why did _person_ go to _place_? </
         template>
</category>
```

Listing 2.1: An example of how NLGML can be used to specify templates for generating questions, taken from Lindberg (2013)

In Listing 2.1, this template defines matching a sentence referring to a person moving to some place. The *place* and *person* entities are extracted and filled-in to the three defined templates.

This question generation system is straightforward to use and implement. However the types of questions generated are limited to those specified within the templates, and information can only be extracted by exact matches to the given patterns. The templating language is not able to generate questions from semantically identical sentences which do not conform to the structure of the template. A sentence such as *James travelled to Japan* is semantically similar to *James went to Japan.* The question templates in Listing 2.1 could be valid for this alternate sentence, but under the given template, no questions would be generated by this system. Overcoming this issue would require massive effort in producing many templates for all semantically similar variations of a given pattern.

2.1.2 Phrase Tree Restructuring

Heilman (2011) describes a fully end-to-end system for generating questions for educational purposes. This rule-based system takes as input some input passage of text and outputs some list of questions generated from extracting clauses from this passage of text and restructuring these clauses into questions through a set of predefined generalist rules. The first stage of this system focuses on clause extraction from longer complex sentences. For example, Heilman argues that given a sentence such as *Prime Minister Vladimir V. Putin, the countrys paramount leader, cut short a trip to Siberia* [sic] that a good question generation system should extract the clause *Prime Minister Vladimir V. Putin cut short a trip to Siberia*, to allow for this to be transformed into a suitable question.

Heilman's described system removes adjuncts and discourse markers using Tregex (Levy and Andrew, 2006) rules which match patterns in phrase structure trees. These matched structures can then be removed or modified as appropriate, with the aim of leaving behind simplified clauses. As described by Heilman (2011), the system removes:

- appositives offset by commas;
- relative clauses offset by commas;
- parentheticals;
- participial modifiers of noun-phrases;
- verb phrase modifiers offset by commas;
- modifiers that precede the subject.

The sentence simplification stage is evaluated with human critics against two baselines. Human critics assessed whether the meaning of an input sentence was preserved following the simplification step. The results suggest that human critics agreed that meaning was preserved following the sentence simplification step with Pearson correlation coefficient $r = 0.82$. The authors find that most mistakes occur as a result of incorrect parsing causing incorrect application of simplification rules. In some cases, extracting simplified clauses can also cause questions with ambiguous subjects to be generated Heilman (2011).

The second stage of this described system takes declarative clauses from the first stage and outputs a set of questions. Question generation is performed using *wh-* movement rules, meaning that *wh-* type questions can be formed with the following words: *who, what, where, whose* and *how many.* To generate potential questions, candidate answer phrases are chosen from the simplified input sentences. Potential answer phrases can be one of noun phrases, prepositional phrases, or subordinate clauses (Heilman, 2011). Following this, the system attempts to identify the type of question to generate based on attributes of the answer phrase. The system uses a supersense tagger to map the answer phrases to a small set of high level semantic classes, such as *PLACE* or *PERSON*. Based on the attributes assigned by the supersense tagger the type of *wh-* word is chosen. For example for the *wh-* word *who*, the answer phrase head word must be tagged as *PERSON*, or must be a personal pronoun. The system has similar rules defined for each of the *wh-* words. Heilman notes that although these rules have some effectiveness, these rules do not allow for choosing the correct *wh-* word in every case. Heilman identifies this area as a topic of possible future research.

For 'yes-no' type questions, subject-auxillary inversion is applied. This is the process of reversing the order of the subject and auxillary verb in a sentence. For example given the sentence *John had eaten the pizza* a subject-auxillary inversion would change the structure of this sentence to: *Had John eaten the pizza.* For sentences without an auxillary preceding the main verb, a verb decomposition step is applied. This step conjugates *do* into the appropriate tense and form to match the form of the main verb using Tregex rules to manipulate the phrase structure tree. The main verb is decomposed into its base form. Heilman presents the example that the sentence *John saw Mary* would become *John did see Mary* following the main verb decomposition step.

A final post-processing step is applied to all generated questions which performs proper formatting such as capitalisation of the first word, and adding the final question mark at the end of the sentence.

The Heilman system applies a ranking system to surface higher quality questions and bury those which are vague. Heilman uses a statistical ranking system based on features of the generated questions such as length, *wh*-words used, negation, and other more general grammatical features.

One drawback to the system described in Heilman (2011) is that questions can only be generated on extractive portions of the input text. In fact, input text is split in sentences before question generation. Therefore, questions can only concern and contain the information of one sentence, rather than consolidating information alluded across sentences and producing questions based on this.

The sentence simplification system aims to extract simple clauses which can be reformulated into questions. While this has been shown in Heilman (2011) to produce simpler questions, the approach can lead to semantically identical questions being generated with and without adjuncts adding extra detail. For example consider this sentence describing a place:

It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858.

Heilman's program will produce both:

1. What is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858?
2. What is it a replica of where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858?

These questions are almost semantically identical, with the only difference being the inclusion of the noun phrase "the grotto at Lourdes, France".

2.2 Neural Question Generation

An alternative approach to rule-based question generation systems are statistical approaches using machine learning techniques.

Serban et al. (2016) present a method using neural network techniques to generate questions from triple tuples of structured 'facts' extracted from a knowledge base. This model is based on neural machine translation models to transduce from structured fact tuples to questions. The model takes as input structured facts, which are a triple of subject, relationship, and object (*subject*, *relationship*, *object*) of some entities. These facts are derived from Freebase, a widely used knowledge base Bollacker et al. (2008). The facts are encoded to vector space embeddings using the TransE model, which has been shown to perform well in link prediction on relational data (Bordes et al., 2013). A fact embedding is a concatenation of the subject, relationship, and object embeddings. These embeddings are then decoded by a recurrent neural network into a sequence of one hot vectors corresponding to the vocabulary words.

Serban et al. (2016) find that this statistical approach compares favourably to a template based baseline on lexical and semantic level evaluations.

A more recent approach to statistical question generation has learned questions directly from source sentences (Du et al., 2017). This approach uses an encoder-decoder recurrent neural network model to encode source sentences and decode these into questions. Du et al. (2017) map questions to source sentences in the SQuAD dataset, and learn to generate questions directly from individual sentences. This approach is in contrast to that used in Heilman and Smith (2010) which generates and ranks questions over a complete article extract.

The neural network architecture used by Du et al. (2017) is similar to other sequence-to-sequence models used in machine translation or automatic summarisation. The tasks share similarities of accepting time-series sequences from one vocabulary and returning sequences in another vocabulary. Intuitively one can reason that the vocabulary of generated questions will have significant lexical overlap with the vocabulary of the source material from which they are formed.

The architecture uses LSTM cells on the encoder and decoder layers and the decoder component uses the attention mechanism described by Bahdanau et al. (2014). The model uses a bi-directional encoder, which reads inputs in sequence and reverse sequence.

The results of Du et al. (2017) suggest that their system is competitive with rule-based overgenerate-and-rank systems (such as those described by Heilman (2011)). Du et al. (2017) evaluate generated questions based on lexical overlap to 'gold-standard' questions, and use human annotators to measure generated question fluency. On measures of lexical overlap and fluency, their model significantly outperforms the rule-based system from Heilman and Smith (2010).

Handling out-of-vocabulary words is a particular challenge for many sequence-to-sequence neural models. In the architecture described by Du et al. (2017), there is no mechanism to generate out-of-vocabulary tokens from texts which are in a different domain to those of the training set. Instead, Du et al. (2017) use a naive approach to handling out-of-domain tokens, which is to replace "unknown" token placeholders with the token from the source text with the highest weighted attention at the time-step of generating "unknown".

Zhou et al. (2017) propose a similar approach to neural question generation. A recurrent neural network encoder-decoder model is trained over sentence-question pairs from the SQuAD dataset (Rajpurkar et al., 2016). Zhou et al. (2017) experiment with using additional features beyond words, such as part of speech tags and named entities from the original text. Gated Recurrent Units (GRUs) are used in place of the LSTM units used by Du et al. (2017). These have been shown to have similar performance to LSTM units and are conceptually simpler Chung et al. (2014). Zhou et al. (2017) also use a more sophisticated method for handling out-of-vocabulary words. Their model uses the method described by Gulcehre et al. (2016) which allows the model to "point" to tokens within the source sentence to include in the generated sequence, in cases where these tokens may not be included in the training vocabulary.

The results suggest that human annotators prefer the questions generated by their neural model over those generated by the rule-based system from Heilman and Smith (2009), on an evaluation rubric set out by Zhou et al. (2017). In the ablation results presented, they find that inclusion of richer features (POS tags and labeled named entities) significantly improves the performance of the question generation model on measures of lexical overlap with gold standard questions. Their argument that as the SQuAD dataset is sourced from Wikipedia articles, it is likely to feature many named entities appears to be sensible from their results. They also observe from qualitative analysis of a sample of generated questions that the system is able to 'copy' spans from the original source sentence, which is useful for handling out-of-vocabulary tokens.

The neural model described by Yuan et al. (2017) also uses an encoder-decoder system. This model generates questions from the input of a document and an answer phrase, where answer phrases are extractive of the document. Sequence elements for the answer and document are fixed during training as GloVe Pennington et al. (2014) embeddings.

A notable difference between the model used by Yuan et al. (2017) and the other neural approaches is the inclusion of a reinforcement learning (RL) component performing Policy Gradient Optimisation. The authors argue that the inclusion of " properly designed reward, maximised via RL, could provide a model with more information about how to distribute probability mass among sequences that do not occur in the training set". The rewards they include as part of the RL component include the accuracy of whether a question-answering system is able to answer a generated question. The authors note that since this may encourage the system to prefer questions which are obvious (such as those containing the answer within the question) they account for this by masking answer words from the original document during generation.

The reinforcement model also rewards minimising negative perplexity of the generated question as a proxy measurement for question 'fluency' (this term is not defined in the paper). The perplexity is measured using a language model trained over the SQuAD dataset.

The model described in Yuan et al. (2017) is evaluated against a baseline sequence-to-sequence model. Their model is found to perform significantly better than the baseline on measures of lexical similarity to gold standard questions and perplexity over the trained language model. In measuring lexical overlap with gold-standard questions the authors use the BLEU (Papineni et al., 2002) and F1 metrics.

These three neural question generation methods present slight differences to their approach. Du et al. (2017) present the simplest system, their approach uses only the input words as features and does not perform Policy Gradient Optimisation as part of the learning strategy. In Zhou et al. (2017), a similar network architecture is used, but with a more sophisticated mechanism for handling out-of-vocabulary words, using the 'copy mechanism' described in Gulcehre et al. (2016). The results from Zhou et al. (2017) also suggest that the fluency and lexical overlap with gold standard questions can be improved by enriching the feature set with part-of-speech tags and tagged named entities. The model described by Yuan et al. (2017) does not include the feature enrichment from Zhou et al. (2017), but does include policy gradient optimisation using a reinforcement learning component which rewards fluency and answerable questions. Using a combination of these two rewards significantly increases the fluency and lexical overlap with gold standard questions over the baseline.

All three approaches use the BLEU-4 (a weighted average measuring n-gram lexical overlap for $n < 5$) metric for identifying lexical overlap between their generated questions and gold-standard questions. Zhou et al. (2017) achieve the highest BLEU-4 score among the papers, although because I do not have access to the raw calculations over the entire dataset, it is not possible to say whether this represents a significant increase. Unfortunately as the evaluation metrics for 'fluency' and rubrics for human quality assessment across these papers are not standardised, further direct comparison is not possible.

2.3 Identifying Salient Sections of Text

In rule-based approaches to generating questions from text, a final ranking step is often included to surface higher quality questions based on lexical features. In experiments by Heilman and Smith (2010) the ranking stage has been shown to correlate with human evaluators scoring of generated questions on a given rubric. This approach is introduced for surfacing the highest quality questions generated from a passage of text, in a system where questions are overgenerated (Heilman and Smith, 2009).

However, this approach may not be the most suitable for systems which generate questions for new question-answering datasets, or in cases where questions are generated on text for language learners. In generating questions for question-answering datasets, it may be more useful to provide a mechanism to select to what extent a system should generate questions on the more salient portions of text. In the case of language learners, asking questions about the most important concepts of a text is likely to be more useful than generating questions on small details of the text, which could be understood without a broader comprehension. For this reason I look at identifying the salient ideas of text for the purposes of question generation.

In Nenkova et al. (2011) the task of summarisation is to "produce a concise and fluent summary of the most important information". Two forms of summarisation are discussed in the literature: extractive and abstractive summarisation. Extractive summarisation selects some subsequence of words from a text and concatenates them together without modification. Abstractive summarisation creates novel sentences which denote the most important information from the text. Additionally within the literature a distinction is made between *indicative* summarisation and *informative* summarisation. Indicative summarisation yields contextual information on the text, for example source and writing style. Informative summarisation instead presents content of the original source text. Within this project, I focus on informative summarisation.

Early approaches to summarisation focused on extractive techniques, involving identifying ‘significant’ sentences. Luhn (1958) argued that sentence significance derived from the significance of its words, and that the significance of words was related to word frequency. That is, words which appear more frequently in a given article are likely to denote subject matter. Therefore, sentences which contain high frequencies of ‘significant’ words are more likely to be salient in a passage of text as a whole. In the approach used by Luhn (1958), this method was used to automatically generate abstracts for technical papers and magazine articles.

A more recent approach to automatic extractive summarisation is a graph-based approach. TextRank (Mihalcea and Tarau, 2004) is one such method that models text units as a graph and ranks vertices according to their importance in the graph. Text units can be individual keywords or sentences - for the purpose of this project, I use full sentences. TextRank models edges between sentences in terms of their ‘similarity’ score to one another. This is calculated as the result of some similarity function between pairs of sentences. The similarity function can be replaced to evaluate different metrics of two given sentences, for example lexical overlap or semantic similarity. For example as given in Mihalcea and Tarau (2004), a lexical overlap similarity measurement between two sentences $S = (w_1 ... w_n)$ and $S_j = (w_1 ... w_m)$ is given in Equation 2.1.

$$Sim(S_i, S_j) = \frac{|\{w_k : w_k \in S_i \wedge w_k \in S_j\}|}{\log n + \log m} \tag{2.1}$$

In generating abstractive summaries, neural network models currently produce state of the art results (Rush et al., 2015). In Rush et al. (2015), the authors present the NAMAS model for producing abstractive summaries of sentences. The model encodes a sequential ordering of words to a real-valued vector space and decodes this vector into a new sequence of words. To evaluate their system, the authors measure the lexical overlap of generated summaries with ‘gold-standard’ summaries over an unseen test set. Additionally as a proxy measurement for fluency of the generated summaries, the authors also include perplexity of the generated summaries, using a language model trained over the entire original dataset.

One strength of abstractive summarisation is that the vocabulary of generated summaries can be different from that of the source text. In the below generated summary example given in Rush et al. (2015), the word ‘against’ features in the generated summary, but is not featured in the original source sentence.

> **source**: russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism
> **summary**: russia calls for joint front against terrorism

However, one major weakness with neural summarisation systems is weak performance on out-of-vocabulary words. For words not seen during training, the network cannot encode such words as a one-hot vector and so it must be replaced by a generic ‘unknown’ token. This can cause poor summaries to be generated, such as with repeated ‘unknown’ tokens or other repeated words and punctuation.

2.4 Evaluating Question Generation Systems

The literature on question generation systems does not provide a consistent methodology for evaluating question generation systems. Over the literature there is a range of methods for evaluating questions on a qualitative and quantitative basis.

On a qualitative basis, authors Heilman (2011); Lindberg (2013) primarily make use of human annotators to score the quality of questions on metrics of relevance to text or readability. These scores are then used to calculate coefficients of correlation with existing human generated ‘gold-standard’ questions.

Heilman and Smith (2009) state that there is not a historically accepted means of evaluating question quality from literature. In this paper, the authors use the ‘precision-at-N’ method for evaluating generated and ranked questions, a commonly used metric for evaluating information retrieval and ranking tasks. Precision-at-n for evaluating question ranking measures the proportion of questions within the top n results which are deemed acceptable by human raters. Raters are given a rubric by the authors for evaluating the quality of questions.

Questions are evaluated by human annotators on the following properties:

- Ungrammatical
- Does not make sense
- Vague
- Too Easy
- Missing Answer
- Wrong wh- word
- Formatting

Chapter 3

Approach

In this section I detail the contributions I have made to the question generation pipeline used in Heilman (2011). I provide a system for automatically learning the type of question to generate, and additional components to identify salient portions of input text. The components I describe extend an existing Java question generation pipeline.[1]

3.1 Identifying Salient Portions of Text

Previous approaches to automatic generation frequently involve a final ranking step based on features of the generated questions, trained on ranking of human evaluators scoring on a range of criteria. Previous ranking approaches have included features such as question length, whether the question contains negation, and other syntactic properties. Ranking based on these features has been shown to correlate well with the human scorers. However, prior approaches do not account for asking questions on the most important portions of a given text, which I hypothesise to be more directly relevant to language learners. This means that questions are ranked according to their quality in a singular context, rather than in the context of the text they are generated

[1]`https://www.cs.cmu.edu/~ark/mheilman/questions/`

from. Similarly for generation of new question-answering datasets, specifying that questions should concern the most salient sections of text would also be useful.

In contrast to this approach, I look at identifying salient portions of text, on which I then apply traditional question generation techniques.

For the purposes of this experiment, the task of textual summarisation is identical to the task of identifying salient portions of text: given textual input, the goal is to produce an abridged version of this text which contains the most relevant portions.

In defining the task, I use similar definitions as those in Rush et al. (2015). I define the input sequence of words as $W = (w_1, ..., w_n)$, and all words come from a vocabulary V and $|V| = v$. I aim to produce a program which takes W as input and outputs a sequence $W_s = (w_1, ..., w_k)$ where $k < n$.

3.1.1 Extractive Approach

The task of extractive summarisation is to find the optimal concatenated subsequences of words from W that maximise some scoring function $S : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.

For some sequence of indices $m = (m_1, ...m_j) : m_i \in \mathbb{N}_{<n+1}$, let $W_{[m]} = (w_{m_1}...w_{m_j})$, that is, the sequence of concatenated words from indices in m. Therefore the task of extractive summarisation is to find some $\bar{m}$ as in Equation 3.1.

$$\bar{m} = \underset{m \in \{1..n\}^n}{\arg\max}\, S(W, W_{[m]}) \tag{3.1}$$

Term Frequency Scoring

I first look at extractive summarisation techniques to identify salient portions of text. The first approach ranks sentences by the length normalised sum of tf-idf scores for each token in the sentence. This approach is derived from the approach used in Seki (2002), the position weighting scheme removed.

For a given term t and document d, the term frequency $tf_{t,d} = f_{t,d}$ is the number of times that t appears in d.

The inverse document frequency $idf_{t,d}$ measures how much information this term provides. For a corpus of documents $|D| = N$, the calculation for $idf_{t,d}$ is given in Equation 3.2.

$$idf_{t,D} = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{3.2}$$

$$tfidf_{t,D} = tf_{t,d} \cdot idf_{t,D} \tag{3.3}$$

The total score for a sentence $s = (w_1 ... w_m)$ in document d is given in Equation 3.4. The score for a sentence is the length normalised sum of tf-idf scores for all terms within the sentence.

$$Score_s = \frac{\sum_{w \in s} tfidf_{w,d}}{m} \tag{3.4}$$

In generating extractive summaries, I select the top 20% of sentences from the original source text. This figure is derived from looking at the DeepMind Read and Comprehend dataset (Hermann et al., 2015), where abstractive summaries of articles are on average close to 20% of the length of the original source text.

TextRank

The second extractive approach I apply uses the TextRank algorithm (Mihalcea and Tarau, 2004). I use the lexical overlap similarity measurement as in Equation 2.1. The score of a sentence is the total similarity score to all other sentences in the document. For a sentence $s = (w_1 ... w_m)$ and document d, the score of s is given in Equation 3.5.

$$Score_s = \sum_{s_i \in d : s_i \neq s} Sim(s, s_i) \tag{3.5}$$

I then sort the sentences by descending score and choose the number of sentences to fit the target summary size.

I use a Python implementation of TextRank.[2]

3.1.2 Abstractive Approach

Where extractive summarisation is the task of selecting a subset of salient sentences from the original text and presenting these in sequence, abstractive summarisation generates new sentences based on information contained within the original text. In this project I also use a neural abstractive approach to identifying salient portions of text.

Neural network approaches have been shown to produce state of the art results for abstractive summarisation. In the experiments on abstractive summarisation I use a slightly modified re-implementation of a state of the art model based on NAMAS (Rush et al., 2015).

The model used in this experiment is a sequence-to-sequence encoder-decoder recurrent neural network. There are two encoder and two decoder layers of 600 LSTM units. The encoded vector is 600 dimensional. The model has been trained over 11 epochs on a single GPU at 346 minutes per epoch - approximately 75 hours of total training time. This model has been trained on the DUC2003, DUC2004 and GigaWord datasets.

I use the OpenNMT framework to build the neural network.

[2] `https://github.com/summanlp/textrank`

3.2 Learning to Choose Wh- Words

One of the steps involved in generating a question is choosing the *wh-* question type. For example, take a sentence such as:

The umbrella is on the beach.

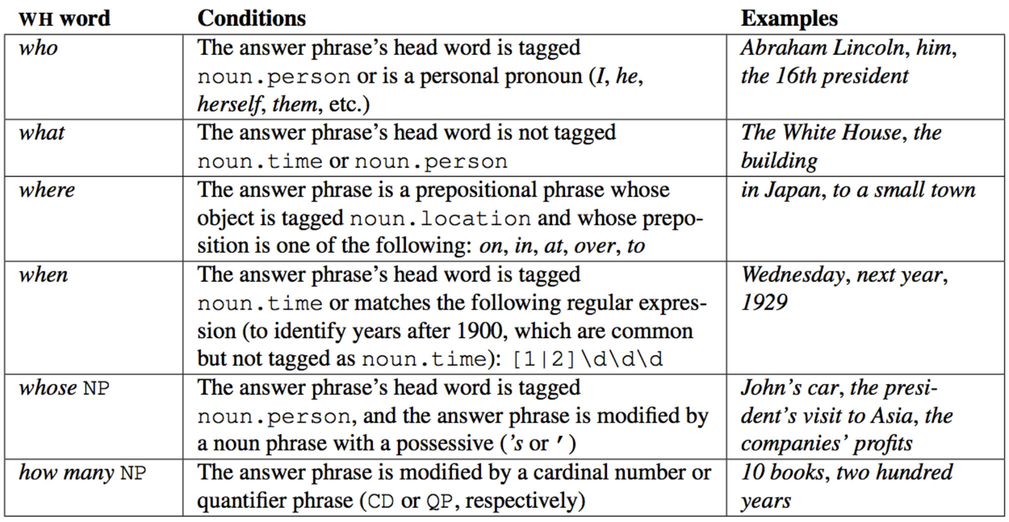
A question that asks for or about the location of the umbrella would be of type “where”, but a question asking the name of the object on the beach would be of type “what”. The choice of *wh-* word is dependent on the answer phrase.

One such approach is to combine knowledge gained from a supersense tagger and morphology. This approach used by Heilman (2011) uses the supersense tagger from Ciaramita and Altun (2006) and several rules to match supersense types and regex patterns in answer phrases to discern which question type to generate. Supersense tagging is the task of assigning higher level semantic categories to words. The supersense tagger used by Ciaramita and Altun (2006) is based on a perceptron trained Hidden Markov Model. This supersense tagger attempts to classify a given entity into one of 26 supersenses for nouns and one of 15 supersenses for verbs.

During the process of generating questions, the answer phrase is assigned a higher level semantic category by the supersense tagger and based on rules defined in Table 3.1, the corresponding *wh-* word is chosen.

My approach aims to directly learn the *wh-* word for a generated question, given an answer phrase. This has the potential to both improve overall performance of the *wh-* word selection task and to remove the manually defined rules.

This task shares some overlap with the supersense tagging task and named entity recognition tasks. However, the named entity recognition task seeks to both identify real-world objects and classify them into high-level semantic categories. In this task, I am not concerned with identifying potential answer phrases, and use only the existing phrase structure tree pattern matching



WH word	Conditions	Examples
who	The answer phrase's head word is tagged `noun.person` or is a personal pronoun (*I*, *he*, *herself*, *them*, etc.)	*Abraham Lincoln, him, the 16th president*
what	The answer phrase's head word is not tagged `noun.time` or `noun.person`	*The White House, the building*
where	The answer phrase is a prepositional phrase whose object is tagged `noun.location` and whose preposition is one of the following: *on*, *in*, *at*, *over*, *to*	*in Japan, to a small town*
when	The answer phrase's head word is tagged `noun.time` or matches the following regular expression (to identify years after 1900, which are common but not tagged as `noun.time`): `[1|2]\d\d\d`	*Wednesday, next year, 1929*
whose NP	The answer phrase's head word is tagged `noun.person`, and the answer phrase is modified by a noun phrase with a possessive (*'s* or *'*)	*John's car, the president's visit to Asia, the companies' profits*
how many NP	The answer phrase is modified by a cardinal number or quantifier phrase (CD or QP, respectively)	*10 books, two hundred years*

Figure 3.1: Rules used by Heilman (2011) to select wh- question type.

WordNet Supersense	*Wh*- word
Noun.Person	Who
Noun.Time	When
Noun.Location	Where
All other Noun Supersenses	What
Verb and Adjective Supersenses	None

Table 3.1: Mappings used from WordNet Supersenses to Wh- word categories

rules used by Heilman (2011). Secondly, the scope of potential answer phrases identified is larger than just named entities. Any noun phrase matched according to the rules from Heilman (2011) within the larger phrase structure tree can be considered a potential answer phrase.

While this task is similar to supersense tagging, there are some notable differences. Instead of classifying entities into over two dozen high level semantic categories of nouns and verbs, I narrow these classes to a subset which corresponds to a *wh*- word. In general this means mapping person senses to 'who', time senses to 'when', location senses to 'where' and all other noun senses to 'what'. I remove the adjective and verb senses from classification, as these cannot form answer phrases under this system.

This task is analogous to a multi-class classification problem. Previous approaches to the supersense tagging problem showed good results using a multiclass perceptron classifier (Ciaramita and Johnson, 2003). In other experiments, multiclass perceptron classifiers have been shown to produce similar results to SVM classifiers, with further arguments made that perceptrons can be more prone to overfitting and have less tendency to generalise (Collobert and Bengio, 2004).

I treat the *wh-* learning task as a traditional classification problem. First I define a set of document label pairs: $D = \{(d_1, y_1)..., (d_n, y_n)\}$. I define a set of classes $C = \{c_1, ..., c_m\}$ and say for a given document label pair (d_i, y_i), $y_i \in C$. A document d_i is real-valued N-dimensional vector and a label y_i is an indicator vector: $y_i \in \{0, 1\}^{|C|}$.

This task aims to learn a function over D such that for a new sample $(d_i, y_i) \notin D$, $f(d_i) = y_i$.

In these experiments I use support vector machine (SVM) classifiers on this learning task. SVMs are a well established classification model that has state of the art performance in a number of textual classification areas. SVM classifiers aim to find a maximally separating hyperplane between two classes of instances. For training vectors in a high dimensional space, an SVM finds some hyperplane with maximal minimum margin - that is, for points within minimal distance to the hyperplane, the distance between the points and the plane is maximised.

SVMs are binary classifiers, but in aggregate can be used on multi-class problems as well, using a 'one-vs-rest' approach. For each class I train a new SVM which learns to differentiate between instances of this class and instances not of this class, i.e: I train m classifiers which will give m results. For predicting the label of a new instance d_k, I can apply a few straightforward rules. In the case where one classifier predicts a positive result, the prediction is the class associated with this classifier. In the case where multiple classifiers predict a positive result, I choose the class of the classifier which predicts with maximal distance to the separating hyperplane. If no classifier returns a positive result, I choose the class of the classifier for which there is minimal distance to the hyperplane.

In the experiments to learn answer phrase to *wh-* word mappings, I use a number of features inspired by previous work on supersense tagging and named entity recognition.

3.2.1 Dataset

The dataset used in this task is the STREUSLE 3.0 dataset.[3] This dataset is a collection of sentences with words tagged with 'supersenses'. The dataset is sourced from the 'reviews' section of the English Web Treebank. Human annotators have manually tagged all single and multi-word supersenses within this collection according to a set of annotation guidelines.[4] The reviews dataset tagged consists of 3,812 sentences, with a total of 8,887 single or multiword expressions tagged with noun supersenses.

The class labels used to tag supersenses are the WordNet supersenses. There are 26 noun supersenses, and 90 other supersenses for verbs, prepositions, and adjectives. Of the noun supersenses which are most relevant to this task, the frequency counts of each type are included in Table 3.2.

An example raw sentence from the dataset is included below and the schema describing the formatting of the dataset is provided at this link.[5]

1 I i PRP O 0 ewtb.r.007403.7
2 am be VBP O-'a 0 'a ewtb.r.007403.7
3 a a DT O 0 ewtb.r.007403.7
4 preferred prefer VBN O-'j 0 'j ewtb.r.007403.7
5 provider provider NN O-PERSON 0 PERSON ewtb.r.007403.7
6 with with IN O-ProfessionalAspect 0 ProfessionalAspect ewtb.r.007403.7

[3]`https://www.cs.cmu.edu/~ark/LexSem/`
[4]`https://github.com/nschneid/nanni/wiki/MWE-Annotation-Guidelines`
[5]`https://www.cs.cmu.edu/~ark/LexSem/streusle-3.0/README.md`

88	ANIMAL
999	ARTIFACT
209	ATTRIBUTE
715	ACT
749	COGNITION
424	COMMUNICATION
88	BODY
427	EVENT
38	FEELING
773	FOOD
1489	GROUP
638	LOCATION
25	MOTIVE
54	NATURAL OBJECT
2	OTHER
1224	PERSON
23	PHENOMENON
5	PLANT
351	POSSESSION
27	PROCESS
110	QUANTITY
35	RELATION
6	SHAPE
52	STATE
34	SUBSTANCE
527	TIME

Table 3.2: Frequency counts of noun supersenses in STREUSLE 3.0 dataset

7 most most JJS O 0 ewtb.r.007403.7

8 insurance insurance NN O-POSSESSION 0 POSSESSION ewtb.r.007403.7

9 companies company NNS O-GROUP 0 GROUP ewtb.r.007403.7

10 . . . O 0 ewtb.r.007403.7

3.2.2 Implementation

I divided the documents into an 80:20 training to test split. For each document in each split, these were then processed to extract all noun supersenses and the associated features. For each noun supersense I used the mapping in Table 3.1 to reduce the 26 noun supersenses into the four *wh-* word classes.

I also provide the performance of the existing "supersense tagger + rules" approach as a comparison to the 'learned *wh-* words' system. In implementing this, I ran the supersense tagger used by Heilman (2011) over all tagged noun supersenses within the test data. I then applied the same rules from the question generation program to assign each of these word expressions an associated *wh-* word.

My *wh-* word assignment system will only choose between a subset of all the *wh-* words used in Heilman (2011). My system will only choose between *who*, *what*, *when*, and *where*, whereas Heilman (2011) also includes *how many* and *whose*. The reason for choosing this approach is that the methodology used by Heilman (2011) to choose *wh-* word *whose* or *how many* only uses rules based on basic lexical features of the answer phrase rather than any assigned supersense (e.g.: *how many* checks that the answer phrase is a number using a regular expression pattern matcher). This means that this learned system could easily be augmented by checking for these lexical characteristics prior to other classification.

The features chosen for this classification task are chosen based on existing features for similar tasks of named entity recognition and supersense tagging. The part-of-speech tag features and 'other lexical features' listed below are all based on features from the supersense tagger in (Attardi et al., 2010).

In the examples below let an answer phrase $A = (a_1, ..., a_n)$ be a sequence of words from a document $D = (d_1, ..., d_m)$.

Contextual Unigrams

I use colocated unigrams from either side of the candidate answer phrase. That is, for a document $D = (d_1, ...d_i, a_1, ...a_n, d_j, ...d_n)$, I include the tokens d_i and d_j as features for classifying A.

Contextual PoS Tags

I use the part-of-speech tags of the directly surrounding words, as parsed by the Stanford Parser using the English PCFG ((Probabilistic Context Free Grammar). That is, for a document $D = (d_1, ...d_i, a_1, ...a_n, d_j, ...d_n)$, I use the part of speech tags for d_i and d_j as features.

The part-of-speech tagset used is the Penn Treebank tagset.[6]

Other Lexical Features

A binary feature indicating whether the answer phrase:

- contains a number. For example, the multi-word expression *3rd annual country run*, would include this feature.
- does not contain alphabetic characters. For example the multi-word expression *321-323-4582* would include this feature.
- is fully capitalised. The single word expression *USAA* would include this feature.
- contains any capitalisation. The multi word expression *United States Census Bureau* would include this feature.
- contains one or more commas.
- contains one or more backslashes.
- contains one or more full stops.

[6] `http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

- contains one or more dollar signs.
- contains one or more colon characters.

I also include a feature indicating the number of tokens within the answer phrase.

3.2.3 Results

I evaluate three models with different types of features. The **LEX** model includes all lexical features listed above; **POS** adds contextual part-of-speech tags and **+CON** adds the contextual words as features.

Features	Class	Precision	Recall	F-score
Lex	where	0.21	0.5	0.3
	what	0.87	0.08	0.15
	who	0.44	0.04	0.08
	when	0.07	0.83	0.12
	avg	**0.71**	0.15	0.15
+POS	where	0.18	0.63	0.28
	what	0.82	0.55	0.66
	who	0.31	0.41	0.35
	when	0.4	0.32	0.35
	avg	0.67	0.52	0.56
+CON	where	0.37	0.41	0.39
	what	0.78	0.81	0.79
	who	0.29	0.27	0.28
	when	0.45	0.27	0.34
	avg	0.66	**0.66**	**0.66**

Table 3.3: SVM classifier results

In Table 3.3 I present the results for my *wh-* word choosing system. From these results it can be seen that model only includes lexical features of the word expression has the highest average precision at 0.71. However the recall of the *what* and *who* classes is less than 0.1. Both the addition of part-of-speech features and the contextual word features cause a significant increase in the F1 score of the model, at the cost of slightly reduced precision on the

Class	Precision	Recall	F-score
where	1.0	0.04	0.07
what	0.73	1.0	0.84
who	1.0	0.1	0.17
when	1.0	0.02	0.04
avg	0.81	0.74	0.64

Table 3.4: Existing SS Approach used by Heilman (2011)

what class, and improved performance on *where* and *when* classes for +POS and +CON models.

Chapter 4

Evaluation

4.1 SQuAD Dataset

The Stanford Question-Answering dataset is a corpus of reading comprehension texts, and questions and answers on these texts. This dataset is primarily designed to test question-answering systems, though I use this dataset for evaluating generated questions against a set of gold standards.

The corpus contains over 500 articles sourced from Wikipedia and contains over 100,000 question-answer pairs on these articles. The articles are divided by paragraphs with an approximate average length of 100 words. Each paragraph has several questions for which 'gold standard' answers are textual segments of the source paragraph. A portion of questions are semantically identical to others from other paragraphs in the same article. For each question, up to five correct answer phrases may be included. These answer phrasings are always semantically identical, but in some cases are alternate representations of the correct answer: for example "19th century" vs "1900s". In cases where there are not alternative representations of the same answer, the same answer phrase is repeated three times.

An example paragraph and list of question-answer pairs from the SQuAD dataset is given below.

The English name ”Normans” comes from the French words Normans/Normanz, plural of Normant, modern French normand, which is itself borrowed from Old Low Franconian Nortmann ”Northman” or directly from Old Norse Normar, Latinized variously as Nortmannus, Normannus, or Nordmannus (recorded in Medieval Latin, 9th century) to mean ”Norseman, Viking”.

In the questions below, gold standard answer phrases are separated by semicolons.

- **Question** What is the original meaning of the word Norman?
- **Answer** Viking; Norseman, Viking; Norseman, Viking;
- **Question** When was the Latin version of the word Norman first recorded?
- **Answer** 9th century; 9th century; 9th century;

Evaluating the quality of generated questions on a given passage of text is challenging. I aim to evaluate generated questions on three dimensions: lexical overlap with gold standard questions; semantic similarity to gold standard questions; and grammatical correctness.

The SQuAD dataset is split into a training, development and test partitions, though only the training and development portions are available online. The test set is withheld to allow for automatic evaluation of question-answering systems online. I evaluate all systems on the development split of the SQuAD dataset, to allow for more direct comparison with other question generation experiments which do the same (Du et al., 2017; Yuan et al., 2017; Zhou et al., 2017).

Each question is assigned a unique identifier. All answers include the word index of their location within the ‘context’ paragraph under the ‘answer_start’ key. The SQuAD dataset is provided in JSON format and an example extract from the raw data file is included below showing the structure of contextual paragraphs, questions and answers.

```
{
  "context": "On May 21, 2013...",
  "qas": [
    {
      "answers": [
        {
          "answer_start": 3,
          "text": "May 21, 2013"
        },
        ...
       ],
      "question": "When was Levi's Stadium awarded
          the right to host Super Bowl 50?",
      "id": "56be5523acb8001400a5032c"
    },
    ...
}
```

Listing 4.1: Example raw data for a contextual paragraph, questions and answers provided in the original SQuAD dataset.

4.2 Evaluation Metrics

4.2.1 Lexical Overlap

The BLEU metric is a measure of lexical overlap between some reference texts and a candidate text Papineni et al. (2002). Originally BLEU was presented as a means to evaluate machine translations between languages on a large scale in an automated fashion. Coughlin (2003) has found that there is a high level of correlation between the metric and human evaluations of translation quality.

Other than evaluating machine translations, BLEU has also been used to evaluate lexical overlap in other tasks, such as in sentence simplification

(Wubben et al., 2012). Wubben et al. (2012) evaluate candidate simplifications of English Wikipedia against the Simple English Wikipedia as the reference texts.

BLEU uses a modified version of n-gram precision to evaluate a candidate text against a set of reference texts. This can be calculated by counting the number of times a given n-gram from a candidate text appears in a single reference text, adding the maximum of the reference count or candidate count of the n-gram and dividing by the length of the candidate text in words. Using a modified version of precision allows for proper penalising of repeated words in a candidate text, in contrast to an unmodified traditional n-gram precision measure.

For a given text an appropriate maximum n-gram can be chosen, although in practice an often used default is quad-grams[1] and this is also suggested by the author in the original BLEU paper Papineni et al. (2002). Of course, if a candidate or reference text is shorter than four words, then a smaller n-gram should be used instead.

The BLEU metric is then the geometric mean of the modified n-gram precisions multiplied by a sentence brevity penalty.

I use the BLEU metric as defined in Papineni et al. (2002). For a set of reference texts $R = \{r_1 ... r_N\}$, let p_r be the modified n-gram precision of a candidate text against a reference text r. Also let w_r be a weighting of reference texts such that $\sum_{r \in R} w_r = 1$, and BP is a brevity penalty.

$$BLEU = BP * \exp(\sum_{r \in R} w_r \log p_r) \tag{4.1}$$

I also use the brevity penalty as defined in the BLEU paper. This penalty exponentially penalises the score of a given candidate text which is longer than the "best length match". The "best length match" text is the text closest in length to the candidate text. For a candidate text length c and the

[1] `http://www.nltk.org/_modules/nltk/translate/bleu_score.html`

"best length match" reference length r the brevity penalty is:

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{1-r/c}, & \text{if } c \leq r \end{cases} \tag{4.2}$$

BLEU is used as the evaluation metric here to measure lexical overlap between generated questions and the gold standard reference questions for a given passage of text.

Measuring lexical overlap in this way presents several challenges:

1. For a given passage of text there may be multiple valid gold standard and generated questions on different topics of the text. It makes no sense to evaluate the lexical overlap between a candidate question on one topic to a reference question on another.
2. The gold standard questions may not include all possible comprehension style questions for a given section of text.
3. The gold standard questions may be rephrasings - semantically very similar or identical.

To evaluate the performance of question generation systems, I make the assumption that the highest BLEU score between a candidate-reference question pair is likely to be asking a semantically similar question. For each generated question, I calculate the BLEU metric for all reference questions and keep only the highest scoring result. I then take the mean of these results over all generated questions.

Accounting for questions being generated which are valid, but not included within the set of gold standard questions for a given dataset is difficult, and may not be captured by this measurement. Additionally, multiple gold standard questions may be rephrasings of the same question. Using the average-maximum method described above avoids this issue as all generated questions will be automatically paired with the corresponding gold standard question with the highest lexical overlap.

4.2.2 Semantic Similarity

I also aim to evaluate the semantic similarity between generated questions and the given gold standard questions using distributional semantics.

Distributional semantic models aim to map syntactically and semantically similar words to similar real-valued vectors. Based on this idea I use the real valued vectors as a representation framework for the generated and gold standard questions, on which I can calculate similarity measures.

Skipthoughts is a high performing unsupervised neural sentence encoder model Kiros et al. (2015), which builds on word-level distributional semantic encoders. This model uses an encoder-decoder model which given a sentence, aims to learn the colocated sentences. It has been shown that Skipthought vectors (sentences encoded to vectors using the skip-thoughts model) perform well in several tasks including question-type classification and semantic relatedness scoring. Kiros et al. (2015) performed semantic relatedness scoring over the SICK dataset (Marelli et al., 2014) against state of the art models and found that Skipthought vectors performed competitively.

Semantic similarity between two real valued vectors is usually calculated by using cosine similarity (Kiros et al., 2015; Marelli et al., 2014) which measures the angle between them. Cosine similarity between two vectors a and b is calculated as in Equation 4.3.

$$cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} \tag{4.3}$$

Vectors with the same angle have a cosine similarity of 1, orthogonal vectors are 0, and diametrically opposed vectors have a cosine similarity of -1.

Similarly to the method used with lexical similarity above, I seek to find the gold-standard question for which a generated question is most semantically similar. Therefore, I also apply the same average-maximum method described above for measuring the semantic similarity between sets of generated questions and sets of gold-standard questions.

4.2.3 Grammatical Correctness

In addition to semantic similarity and lexical overlap with gold standard questions, the questions themselves must be grammatically correct. While evaluation metrics such as the BLEU score will capture the lexical similarity between a gold standard question and a generated question, they don't take into account how grammatically correct this generated question may be. For example given a gold standard question, a random word-level permutation of this question could score highly on measures of lexical overlap or semantic similarity, but to an average reader the generated question would clearly be nonsensical. For this reason I seek to measure the 'fluency' of generated questions.

One method of showing the fluency of a given text is to measure the perplexity of a set of generated questions using a sufficiently powerful language model. Within the context of a language model, which is just a probability distribution over sequences of words, perplexity measures how well a language model predicts some given text. Assuming the language model is sufficiently representative of real world samples of textual data, perplexity under this language model can be used to tell how likely a given sequence of words might be. This approach requires that I make several assumptions when using this language model. Firstly, I assume that the language model has been trained over a corpus of text which is itself entirely grammatically correct, or that ungrammatical sequences are extremely rare, causing ungrammatical sequences to be assigned low probabilities.

Secondly, I assume that grammatical sequences will be assigned high probabilities. In practice, these assumptions may not be true. As the language model training data is finite, it is possible that grammatical sequences of words are unseen or occur rarely, causing them to be assigned low probabilities (and high perplexity).

Perplexity for a sequence of words $W = (w_1 ... w_N)$ is calculated as in Equation 4.4. Let p_{w_i} be the probability of word at index i. In the language model used by Jozefowicz et al. (2016) this includes the end of sentence symbol.

$$P = e^{-\frac{1}{N}\sum_{i=1}^{N}\ln p_{w_i}} \tag{4.4}$$

This is one approach used by Yuan et al. (2017) in evaluating the quality of generated questions using a neural network model. In this case the authors have trained a language model over the SQuAD dataset and used perplexity as a measure for 'fluency'. One issue with using a language model trained over just the SQuAD dataset is that it restricts the possible sequences of words to those featured in the dataset. Sentences and questions can be written in many ways with little lexical overlap but be semantically identical. For this reason, I use a more general language model in evaluating the generated questions.

I use the language model implementation described by Jozefowicz et al. (2016) which has been trained on the 1 billion word benchmark dataset (Chelba et al., 2013). This dataset has a vocabulary of approximately 800k words and consists of a large collection of news articles from a variety of Internet sites and news commentary from the News Commentary Corpus.[2] The language model described by Jozefowicz et al. (2016) is a neural model and has state of the art perplexity of 30 over the test dataset.

I use the author's implementation of this language model[3] to evaluate the perplexity of generated questions. In calculating perplexity I separate out all generated questions and calculate the perplexity of each question under the language model. I then average these values to give the average perplexity under a model.

While language models are useful for calculating the probability of sequences of words under a distribution, there can be caveats to this approach. Rare words and rare sequences of words can cause grammatical sentences to be given low probabilities. Similarly, ungrammatical sequences can be given high probabilities if they appear frequently in the training data.

[2]http://www.casmacat.eu/corpus/news-commentary.html

[3]https://github.com/tensorflow/models/tree/master/lm_1b

Another approach which could handle these edge cases would be to look at the probability distribution of the syntactic classes of words in questions generated.

Therefore I look at another approach to measuring grammaticality of generated sentences. I also use unlexicalised PCFG parsers to gain a proxy measurement for grammatical correctness. An unlexicalised grammar makes distinction between classes of function words and content words, in that only rules at the level of individual words for closed classes can be included. In Klein and Manning (2003) the authors state that this means rules for PP[to] can be defined, but not rules for NN[stocks]. I use an unlexicalised grammar to gain a simplistic proxy measurement for the extent to which a sentence's syntactic structure is probable.

PCFG grammars include parameters over all the rules in the grammar which are conditional probabilities of choosing this rule given a non-terminal expansion Collins (2013). As such, PCFG parsers can calculate the overall log-probability of a given parse by summing the log-probabilities of all rule applications over a given passage of text.

Length normalisation becomes important when using these techniques. Longer sentences are less likely, as more <1 probabilities are multiplied together. A length normalised probability score for a sequence of words $S = (w_1...w_n)$ can be calculated as follows:

$$p_{length\_norm}(S) = \frac{\log p(S)}{n} \tag{4.5}$$

Using these metrics as proxy measurements is not without caveats. For a given language model, the score can also be affected by the rarity of words in the text being evaluated. Rarer words will increase the perplexity and decrease the probability over a text sample, even if the word ordering is entirely grammatical.

Similarly, syntactic structures which are rarer, or sequences of closed class words which are less probable may be assigned low probabilities despite being grammatically correct. Because of these caveats, the provided evaluations on sentence ‘fluency’ should be interpreted with some reservation.

4.3 Models Evaluated

In Heilman and Smith (2010) and as described in 2.4, a ‘precision-at-N’ method is used to evaluate questions. In the SQuAD dataset there are an average of 5.1 questions per contextual paragraph. Therefore for each set of generated questions in all the below models, I narrow the scope of questions which are evaluated to only the top-5.

4.3.1 Salient Section Identification

No Summ - None

This model does contain a component for identifying salient portions of text. This is the baseline model as described in Heilman (2011).

tf-idf - Highest tf-idf sentences

In this model, I identify salient portions of text by calculating the sum of tf-idf scores within a sentence and selecting the sentences with the highest score. This is the method described in Section 3.1.1.

TextRank

I use the TextRank model to extract salient sentences from the original paragraph. This approach is described in Section 3.1.1.

4.3.2 Choosing Wh- Words

Wh-Rules - Rules Method

In this model I use the existing approach used in Heilman (2011). This involves tagging of answer phrases using a supersense tagger and then choosing the *wh-* word based on the tag chosen and additional syntactic rules. These rules are described in Figure 3.1.

Wh-learn - *Wh-* Word Learner

This model uses the *wh-* word learner component described in Section 3.2.2. The SVM classifier described was trained over the STREUSLE dataset.[4] The Java project used by Heilman (2011) has been modified to make a remote procedure call to an external service which runs the SVM classifier. The classifier returns the *wh-* word to generate. I use the classifier with the highest overall F1-score, from the results this is the classifier using all features described.

4.3.3 Neural Model

As part of this project I also include a re-implementation of the end-to-end neural question generation model used by Du et al. (2017).

This model is an encoder-decoder recurrent neural network with LSTM units and the model described below is based on the description in Du et al. (2017).

The neural network model for generating questions is presented below. The model takes as input a sentence as a sequence of words $x = (w_1 ... w_n)$ and outputs a question as a new sequence of words $q = (v_1 ... v_m)$. This task concerns finding $\bar{q}$ given x as in Equation 4.6.

[4]`https://www.cs.cmu.edu/~ark/LexSem/`

$$\bar{q} = \arg\max_{y} P(q|x) \tag{4.6}$$

Equation 4.6 is factored into products of word predictions as in Equation 4.7, where $q_{<i}$ are the predictions of previous words - the index of these words is less than i.

$$P(q|x) = \prod_{i=1}^{m} P(q_i|x, q_{<i}) \tag{4.7}$$

Let the state variable of the neural network at time t be $\boldsymbol{h}_t$. Also let c_t be the attention based encoding of x at time t (to be elaborated on below). Then $P(q_i|x, q_{<i})$ can be calculated as in Equation 4.8 where $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ are parameters to be learned.

$$P(q_i|x, q_{<i}) = \text{softmax}(\boldsymbol{W}_s \tanh \boldsymbol{W}_t[\boldsymbol{h}_t; \boldsymbol{c}_t]) \tag{4.8}$$

The context vector $\boldsymbol{h}_t$ is calculated using the Long-Short-Term-Memory network (Hochreiter and Schmidhuber, 1997) as in Equation 4.9, generating new state from the previous state $\boldsymbol{h}_{t-1}$ and previous word generated q_{t-1}.

$$\boldsymbol{h}_t = \text{LSTM}(q_{t-1}, \boldsymbol{h}_{t-1}) \tag{4.9}$$

The training of this network seeks to maximise the probability of the output sequence q given some input x. I use a training corpus of sentence-question pairs $P = \{(x^{(i)}, q^{(i)})\}_{i=1}^{p}$. The model training objective is to maximise the likelihood (minimise the log-likelihood $\mathcal{L}$) of the training data with respect to all the parameters of the network, θ. Log-likelihood is calculated as in Equation 4.10

$$\mathcal{L} = -\sum_{i=1}^{p} \log P(\boldsymbol{q}^{(i)}|\boldsymbol{x}^{(i)}; \theta) = -\sum_{i=1}^{p}\sum_{j=1}^{i} \log P(q_j^{(i)}|\boldsymbol{x}^{(i)}, q_{<j}^{(i)}; \theta) \tag{4.10}$$

The LSTM encoder is a bi-directional attention-based encoder. Inputs are fed into the encoder in order and in reverse order. The encoded vector is the concatenation of the backwards and forwards pass encodings. The attention based encoding ($\boldsymbol{c}_t$) is the weighted average over encoding timesteps. Let the bi-directional encoding at timestep t be $\boldsymbol{b}_t$. The attention based encoding $\boldsymbol{c}_t$ is calculated as in Equations 4.11 and 4.12.

$$\boldsymbol{c}_t = \sum_{i=1}^{|\boldsymbol{x}|} a_{i,t}\boldsymbol{b}_i \tag{4.11}$$

$$a_{i,t} = \frac{\exp\left(\boldsymbol{h}_t^T \boldsymbol{W}_b \boldsymbol{b}_i\right)}{\sum_j \exp(\boldsymbol{h}_t^T \boldsymbol{W}_b \boldsymbol{b}_j)} \tag{4.12}$$

In my implementation of this model, I use the OpenNMT framework (Klein et al.). In the source vocabulary I keep 50k of the most frequently used tokens and in the target vocabulary I keep 30k of the most frequently used tokens. All other tokens are replaced by a generic 'unknown' token. I use the *glove.840B.300d* pre-trained embeddings as fixed inputs to the neural network, with a random unit vector used for tokens not included in the GloVe vocabulary. Word embeddings are fixed during training.

For this neural model I used a slightly modified version of the SQuAD dataset. In the original dataset, sets of question-answer pairs were paired with paragraphs. In this modified version, question-answer are now associated with individual sentences containing the information from which the question and answer is sourced. Questions have been automatically paired with sentences from the original paragraphs by selecting the sentence with which the gold-standard question has the highest lexical overlap, after stop-words are removed from both the question and source sentence. I use the pre-processed version of the dataset provided by Du et al. (2017).[5] Some sentences are associated with multiple gold-standard questions, the average number of questions per sentence is 1.4 (Du et al., 2017).

This dataset has been tokenised using the tokenisation module of OpenNMT

[5]`https://github.com/xinyadu/nqg`

and shifted to lowercase.

I use many of the same hyperparameters as in Du et al. (2017). Optimisation is performed with Stochastic Gradient Descent, using a learning rate of 1.0 initially which then halves after 6 epochs. Total training time is approximately 3 hours.

During inference, I use beam search with a beam size of 3 to find the most probable sequence of tokens. Decoding halts after the 'end of sentence' token is generated.

As I generate questions at the sentence level, the evaluation of this model is performed slightly differently as well. For lexical similarity and semantic similarity measurements I do not use the 'average-maximum' approach used to evaluate the other models. Instead I directly measure the lexical overlap and semantic similarity with the known gold-standard question directly, using the BLEU scoring system and the cosine-similarity of the Skipthought vectors. The perplexity scoring and parser scoring is performed in the same way as for the other models.

4.4 Quantitative Evaluation

Model		Lex	Sem-Sim	Grammaticality	
				Perp	Parse Score
Wh-Rules	No Summ	**0.795**	**0.682**	499.5	-15.04
	tf-idf	0.792	0.664	322.2	-15.02
	TextRank	0.789	0.669	304.4	-14.91
	Abstr.	0.719	0.519	420.2	-18.09
Wh-Learn	No-Summ	**0.795**	0.669	389.3	-15.23
	tf-idf	0.791	0.650	462.6	-15.31
	TextRank	0.788	0.654	349.2	-15.33
	Abstr.	0.718	0.499	432.2	-19.18
Neural		0.717	0.539	**68.0**	**-9.78**

Table 4.1: Results of all models evaluated

	Wh-Rules + No-Summ	**Wh-Rules + tfidf**	**Wh-Rules + TextRank**	**Wh-Rules + Abstr**	**Wh-Learn + No-Summ**	**Wh-Learn + tfidf**	**Wh-Learn + TextRank**	**Wh-Learn + Abstr**
Wh-Learn + Abstr	<<	<<	<<	-	<<	<<	<<	x
Wh-Learn + TextRank	-	-	-	>>	-	-	x	
Wh-Learn + tfidf	-	-	-	>>	-	x		
Wh-Learn + No-Summ	-	-	-	>>	x			
Wh-Rules + Abstr	<<	<<	<<	x				
Wh-Rules + Textrank	-	-	x					
Wh-Rules + tfidf	-	x						
Wh-Rules + No-Summ	x							

Table 4.2: Significance levels of **lexical** similarity of generated questions to gold standard questions between models. '>>' indicates the model in the row header performed better than the classifier in the column header at $p < 0.01$. '<<' indicates the model in the column header performed better than the row header at $p < 0.01$. '-' indicates no significant difference.

The main results of my evaluation are presented in Table 4.1. This table presents the results for nine different models evaluated for lexical and semantic similarity to gold standard questions, as well as grammaticality or 'fluency' of generated questions. A full explanation of each evaluation metric used can be found in Section 4.2. Explanations on each of the models evaluated can be found in Section 4.3.

A two-tailed Student's *t-test* was performed between lexical similarity and semantic similarity scores for all models. These results can be found in Tables 4.2 and 4.3.

The best performing models in terms of lexical similarity to the existing gold standard questions are the **Wh-Rules + No-Summ** and **Wh-Learn + No-Summ** models. However, these models are not significantly different

	Wh-Rules + No-Summ	**Wh-Rules + tfidf**	**Wh-Rules + TextRank**	**Wh-Rules + Abstr**	**Wh-Learn + No-Summ**	**Wh-Learn + tfidf**	**Wh-Learn + TextRank**	**Wh-Learn + Abstr**
Wh-Learn + Abstr	<<	<<	<<	-	<<	<<	<<	x
Wh-Learn + TextRank	-	-	-	>>	-	-	x	
Wh-Learn + tfidf	-	-	-	>>	-	x		
Wh-Learn + No-Summ	-	-	-	>>	x			
Wh-Rules + Abstr	<<	<<	<<	x				
Wh-Rules + Textrank	-	-	x					
Wh-Rules + tfidf	-	x						
Wh-Rules + No-Summ	x							

Table 4.3: Significance levels of **semantic** similarity measurements of generated questions to gold standard questions between models. '>>' indicates the model in the row header performed better than the classifier in the column header at $p < 0.01$. '<<' indicates the model in the column header performed better than the row header at $p < 0.01$. '-' indicates no significant difference.

from other models including a component for identifying salient portions of text. One possible reason for this result is that the question ranking method based on lexical features of questions used in Heilman and Smith (2010) already correlates well with additional methods for identification of salient sections of text prior to generation. The models using the neural abstractive summarisation component performed significantly worse than other models, this is likely in part due to the failures of handling tokens unseen in the original training dataset.

In terms of semantic similarity, there is a similar pattern. The **Wh-Rules + No-Summ** model produces questions with the highest semantic similarity to gold-standard questions, though is not significantly different from models including a component identifying salient portions of text. Here again the models including an abstractive summarisation component performed significantly worse than other models.

The neural question generation model is not directly comparable to other models, due to the differences in the dataset being used - the neural model is aligned to gold standard questions at the sentence level, whereas the other models are aligned at the paragraph level.

4.5 Qualitative Evaluation

In some cases quantitative measurements may not capture the full extent to which models perform in generating questions. For example, differences in *wh-* word type would significantly affect the quality of a question for a reader, but this word would only constitute a fraction of a total score, when measuring the full semantic or lexical similarity to a gold-standard question. For this reason I also perform a qualitative evaluation of 100 randomly sampled questions from the baseline **Wh-Rules + No-Summ** model and the **Wh-Learn + TextRank** model.

This qualitative evaluation methodology is derivative of the method used in Heilman and Smith (2010). That is, questions are scored for binary presence of several categories of binary 'failure' features. In cases where generated questions contain none of these features, questions are marked as 'no issues'.

In Heilman and Smith (2010), questions are qualitatively evaluated for several types of question weakness:

- Vague
- Ungrammatical
- Obvious answer
- Missing answer
- Wrong WH word

- Formatting
- Other
- No Issues

From the failure categories used in Heilman and Smith (2010), I combine the categories of formatting, ungrammatical and other question errors under a single "grammatical errors" category. I also separate out the vague category into two further sub-categories of *ambiguous subject* and *ambiguous details*. I do not include the category of *missing answer* as I exclude all questions without answers prior to evaluation. I tabulate the types of errors observed by this system in generating questions below and provide example generated questions for each case.

- **Wrong *Wh-* Word** - *Who was India?*
- **Ambiguous Subject** - *What is the product?*
- **Ambiguous Details** - *What did the West announce at the same time?*
- **Unnecessary Details** - *Who encountered clans of the Herero tribe at Windhoek, Gobabis, and Okahandja which were less accommodating on the missionaries accompanying the Orlams's way further northwards?*
- **Answer Contained in Question** - *What did the Egyptian and Syrian armies launch a surprise attack against Israeli forces in the Sinai Peninsula and Golan Heights on?*
- **Other Grammatical Errors** - *What can works regularly be heard on radio stations on?*
- **No Errors** - *When did Roman Catholic Jesuits and Capuchins arrive from Europe?*

The frequency counts for totalling all types of errors made do not total 100 questions, as some generated questions can feature more than one type of error, and so will be assigned to two categories simultaneously.

Table 4.4 contains the frequency count of error types for the baseline **Wh-**

Type of Error	Baseline Freq.	Wh-Learn + TextRank
Wrong Wh- Word	15	10
Ambiguous Subject	27	12
Ambiguous Details	11	3
Unnecessary Details	5	4
Answer in Question	4	8
Other Gramm. Errors	12	20
No Errors	41	57

Table 4.4: Error Type Frequency for baseline model and **Wh-Learn + TextRank** model

Rules + No-Summ model and the **Wh-Learn + TextRank** model. The most frequent error type observed produced by the baseline model is ambiguous subject. For the **Wh-Learn + TextRank** model, other grammatical errors were the largest category. Within the sample observed, a proportion include an unresolved pronoun as the main subject. Other observed examples of ambiguous subjects would include questions such as *Who is the woman?*. Incorrect wh- word for a given question is the second most frequently observed error type, followed by other grammatical errors, superfluous details and questions containing the answer.

While this simplistic qualitative analysis is not necessarily confirmation of increased performance in reducing overall errors, the results are promising and suggest that performance is at least on a par with the baseline model.

Applying System to Language Learners

I have also applied a model to a sample of texts from a language learner's examination dataset (Xia et al., 2016). These are provided for demonstrative purposes in Appendix A.

Chapter 5

Summary and Conclusions

This project investigated methods for automatically generating factual questions from passages of text. I have added components to an existing project generating factual questions with the aim of improving the quality of questions generated. The first component aims to identify salient portions of text, from which factual questions should be generated. The second component seeks to replace a supersense tagger and rule-based *wh-* word choosing component, with a new learning system for automatically choosing the category of question to generate. I have also re-implemented a recent neural network approach for automatically generating questions from passages of text. I evaluate combinations of these models for lexical and semantic similarity to gold standard questions. I also perform evaluations targeting the 'fluency' and grammaticality of generated questions.

The results suggest that a component learning to choose *wh-* words automatically can perform well in comparison to existing approaches. However, I have not seen a significant improvement in lexical or semantic similarity to gold standard questions from models seeking to identify the salient portions of text. In my experiment, I have made the assumption that gold standard questions within the SQuAD dataset are likely to be sourced from the most salient portions of each contextual section of text. In practice, this assumption may not be true. Despite this, the quality of generated questions under extractive summarisation methods is not significantly different from questions generated without this summarisation step applied to contextual material.

There is potential for significant future work within this area. Within the rules based system modified in this project, further work could be done to improve the performance of the *wh-* word chooser built within this project. Possible improvements could include more lexical features. Given that including just contextual words as features significantly improves the performance of the *wh-* classifier, a larger dataset to train this component on could improve performance.

The neural network approaches discussed in Section 2.2 also present significant potential for future models. One major advantage of these models is that they do not depend on complicated and fixed rules which require expert guidance to create, for generating questions. Instead, all of neural network models discussed in Section 2.2 use only slightly modified generic recurrent neural network implementations. These types of models also have good performance on other tasks such as abstractive summarisation or language translations, suggesting that they are suited for generalising to many types of sequence-to-sequence tasks. An issue with the neural network approaches is that currently they require sentences as input, and are only able to generate one question per sentence. In the work of Du et al. (2017), the authors showed that on average there were 1.4 gold standard questions within the SQuAD dataset per sentence. In cases where it should be useful to extract many types of questions from a complicated input sentence, it may be necessary to apply a pre-processing simplification step to extract multiple sub-clauses from the source sentence. Each of these clauses could then be used as input to the neural network instead.

The results of this project show that the quality of automatic factual question generation depends greatly on the type of models used. The rules based system for manipulating phrase structure trees is viable. Future work remains on neural models as well, moving beyond modifications of generic neural architectures perhaps towards a more specialised system.

Appendix A

Applying System to Language Learner Datasets

Within this section I apply a question generation pipeline to language learner datasets. I choose a random passage from each dataset of PET, KET, FCE, CPE and CAE examinations (Xia et al., 2016) and apply the *Wh-Learn + No-Summ* model to each. I provide the top-3 ranked questions generated. These examples are provided for demonstrative purposes only.

A.1 KET

This examination is a "basic-level" English qualification suitable for demonstrating "your ability to use English to communicate in simple situations".[1]

> People are interested in the weather for a lot of reasons and it is important for people in many different jobs. For example, the weather makes a big difference for farmers. Many years ago, farmers spent a long time looking at the sky and at animals and flowers to try and learn more about the weather. Now there are lots of computer programmes which help us to learn about the

[1]`http://www.cambridgeenglish.org/exams/key/`

> weather, and people can even get weather information on their mobile phones. Today, a lot of people still believe in the old ideas about the weather. They say, for example, that if there is a red sky in the evening, it will be sunny the next day. Some people also think that when cows lie down in the fields, it will soon rain.

Generated Questions:

- What is important for people in many different jobs?
- What makes a big difference for farmers for example? *Unnecessary details
- What are interested in the weather for a lot of reasons? *Incorrect *wh*-word

The incorrect *wh*- word is chosen for the third question, where this has been chosen as *what* it instead should be *who*.

A.2 PET

This examination is to demonstrate "mastery of the basics of English". It is the next step in difficulty after the KET.[2]

> Since it opened, the Eden Project in the south-west of England has become one of the UKs most popular tourist attractions. At first sight, the huge biospheres on this former industrial site look like a scene from a science-fiction movie. Each biosphere contains a different climate zone, showing visitors just how important the relationship between people and plants is. It is amazing to see just how many products that we use every day come from plants: wood, rubber, fruit, rice, sugar, coffee and chocolate, for example. Over 135,000 different types of plant are grown here. However cold the weather is outside, computer-controlled

[2] `http://www.cambridgeenglish.org/exams/preliminary/`

> electric heaters keep the plants at the right temperature. In the Rainforest Zone, the atmosphere is always damp because of a huge waterfall. Another zone reproduces the environment of the Mediterranean, California and South Africa, and there are plans to build a desert zone in the near future.

Generated Questions:

- Who has become one of the UK's most popular tourist attractions? *Incorrect *wh-*
- What has the Eden Project in the south-west of England become?
- What is always damp because of a huge waterfall in the Rainforest Zone?

With the exception of the incorrect *wh-* word chosen for the first question, the others are otherwise of good quality.

A.3 FCE

This examination is to prove a person has "the language skills to live and work independently in an English-speaking country or study on courses taught in English." It is the next difficulty level after PET.[3].

> First impressions are often lasting ones. Studies show that people form impressions about us within the first few minutes of meeting. They observe how we dress, our eye contact, our body movement and how fast or slowly we talk, our volume and tone of voice as well as our our actual words. Mary Pearce studied to be a teacher. She says, 'I worked hard to earn my degree. When I finally graduated I was very confident.' She applied for a job at a nearby primary school and got an interview with the Head Teacher.'I noticed a small hole in my jacket that morning,'

[3]http://www.cambridgeenglish.org/exams/first/

she recalls. 'I would have changed, but I knew it would make me late, and I always think it's important to be on time.' Mary didn't get the job. In fact, one of her friends who also teaches at the school told her the Head Teacher's only comment was, 'If someone doesn't take the time to present her best image at an interview, what kind of teacher is she going to be?' As Simon Grant, hotel manager, says: 'Interviewees who look as if they care about themselves are more likely to care about their jobs. People think it'swhat's inside that counts, but in an interview you should aim to come across in the best possible way.' Yet many people ignore the importance of having a professional image. For example, Janet Goodwood worked for ten years as an administrative assistant in a large accounting firm. When the office manager retired, she applied forthe position but wasn't even given an interview. 'I thought it was a mistake so I asked the Director of Personnel what had happened,' she says. 'He told me I didn't fit the image of an office manager. He suggested I improve my wardrobe before I applied again for promotion. I was shocked. I do a very good job and the way I dress shouldn't make any difference.' Movement and gestures will also influence an interviewer's first impression of a candidate. Psychologist Albert Mehrabian has discovered that 7% of any message about our feelings and attitudes comes from the words we use, 38% from voice and a surprising 55% from our facial expressions. When our facial expressions and our words send different messages the listener will put more weight on the non-verbal message. So make sure your words agree with your body language. Mixed messages will only confuse the interviewer. It is also important not to appear too desperate for the job or too eager to please. When Sheila Rice, a marketing specialist, applied for a promotion her interview went so well she was offered the job on the spot. 'I was delighted,' she recalls. 'But I reacted to the offer with too much enthusiasm. Once the boss sensed how excited I was, he knew I wasn't

> going to turn him down. Consequently, he offered me a lower salary than I'd hoped for. I'm sure I could have got more had I managed to control my excitement.' Finally, a consideration of what we say and how we say it will contribute to the success of an interview. David Artesio, the manager of an employment agency, suggests that it's a good idea to inform yourself about the company before you go for an interview. 'The annual report, for example, will tell you about areas of company involvement. Mention an area that interests you during the interview. This will give a positive note and convince others of your interest in the company.' Business consultant Marian Woodall suggests you have a few questions ready and avoid speaking in long, confused sentences. As she puts it, 'Poor communicators talk in paragraphs. Successful communicators talk in short sentences and even in highlighted points.'

Generated Questions:

- Who got an interview with the Head Teacher?
- What did Janet Goodwood work for ten years as for example?
- Who worked for ten years as an administrative assistant in a large accounting firm for example?

The additional prepositional phrase included in the second question "for example" creates an awkward phrasing, even though the overall meaning remains intact. Similarly for the third question the same issue is observed.

A.4 CAE

This is a high-level qualification to demonstrate "high-level achievement in learning English". It is the next difficulty level after FCE.[4].

[4] `http://www.cambridgeenglish.org/exams/advanced/`

Susan Harr unplugs her gadgets and rediscovers the joys of manual labour. Everyone is in love with technology. It gives us all those marvellous gadgets that make life easier and leave us so much more time to do other things. A gradual, though not particularly subtle, form of brainwashing has persuaded us that technology rules, and that it is OK. These implications are obvious. The movement of my fingers uses nothing from the previous power supply being eaten up by our greedy race. A craft executed by hand does not pollute the environment. However, a recent unhappy experience with my malfunctioning word processor - a 48 call-out fee, a labour charge of 15 per quarter of an hour, plus parts and replacements costs - has confirmed a suspicion that gadgets are often not worth the expense or the trouble. Are we as dependent on technology as we imagine? Bit by bit, I have been letting the household technology fall by the wayside as its natural and often short life expires. This makes me wonder just what 'time' technology gives us. The time to take up more activities for which we must buy more gadgets? If so, hats off to the marketing experts: but I think they are conning us. So when the thing started making curious noises, which continued even when it was disconnected by a puzzled service agent, I abandoned it to the backyard, where it whispers damply to itself like some robot ghost. I am not tied to a noisy, whirring machine, with my head bent and my back turned on the world, and I can take my time over the garment. In any case, I was always slightly alarmed by those electric machines that dash across the fabric towards your fingers. Best of all, I can pop the whole lot into a carrier bag and take it with me wherever I go. Of course, there are some gadgets I would not like to be without. A year living without a washing machine convinced me of the value of the electric washtub. But there are others whose loss has brought unexpected delight. Feeling that we were becoming too apt to collapse in front of the television, or slot in a video, I sent back the rented colour equipment and we returned to the small black-and-white portable. It is a real strain on the eyes and concentrates the mind on what is really worth watching. We now spend a lot more time walking the dog (who never liked television anyway), reading, talking or pursuing other hobbies. One of these, in my own case, is sewing; and here is another gadget that went by the board. My old Singer sewing machine is now an ornamental plant table, and as l cannot afford to replace it, I have taken to sewing by hand. We have come to believe that we could not do without it, and if we do resist the notion that our lives would be unmanageable without the appliances of science we certainly do not want to relinquish them. Pity the generations whose lives were blighted by tedious and blister inducing toil. Even our brains are relieved of exertion by computers that not only perform miraculous calculations with amazing speed but now provide entertainment. In fact, the time I now spend placidly stitching is anything but tedious, and the advantages are numerous. For a start, I can sew and listen to the radio - another rediscovered pleasure - or I can talk with family and friends. If it is a simple task, I can watch the programmes I do want to see on television, and alleviate my puritanical guilt at sitting in front of the box by doing something useful at the same time. And what a lovely, cosy feeling it is to sit by the fire and sew with a pot of tea for company. Quite wrongly, I had tended to think with horror of the women who sewed elaborate garments, robes, linen and household items by hand. I thought of those long hours, the strain on the eyes and so on. There is a wonderfully soothing quality about executing a craft by hand, a great satisfaction in watching one's work become neater, more assured. I find things get done surprisingly quickly, and the pace of life suddenly slows down to the rhythm of my own hands. I am also freed from one of the most detestable aspects of late 20th century life the need to rush to finish an activity so that I can rush to the next. Meanwhile I have regained control of my sink, where I plunge my hands into the suds and daydream while doing the washing up agreeable, if temporarily forgotten, activity. The result of all this brooding is that I now prowl the house with a speculative eye. Do we really need the freezer, the microwave oven, that powered lawnmower? Come to think of it, we could save an awful lot of money by doing without electric lights!

Generated Questions:

- Who unplugs her gadgets and rediscovers the joys of manual labor?
- What convinced me of the value of the electric washtub?
- What could save an awful lot of money by doing without electric lights? *Incorrect *wh-*

The incorrect *wh-* word is chosen for the third question.

A.5 CPE

With her children now grown, widowed Susan faces leaving the family home.

The van said, Susan noticed, Removers of Distinction, and indeed, every distinguishing feature of the house was being removed. Everything which made it particular was being wrapped in newspaper and packed in boxes by Fred the removal man, his enormous fingers like sausages tenderly handling all the breakables, and his team of helpers, not so gentle. It was a lovely house that she was leaving, an elegant four-storeyed building overlooking a tiny harbour. The years she had spent there, the years of the children growing up and leaving, hung around in the air, faintly present like agitated dust. When told that they had bought this house, Robert, then five, had asked thoughtfully, Mum, when you buy a house, how dyou get it home? You could miss a little boy in the physical presence of the adult he had become; Robert was here, helping, and in particular making sure she didnt let on about the piano. Francesca was here too, also helping, in her bossy way, stubbornly certain that nobody but she, the family daughter, would be careful enough over a fine instrument like a Steinway piano. She doesnt look like shes going to cry on us, observed Fred. Thats something. Do people cry? Susan asked, intrigued. You'd be surprised, said Fred. They go around merry as magpies helping out till its all in the van, then you look round and there they are, crying in the middle of an empty room. They're fine when we get to the new place, mind. It's just seeing everything taken apart that upsets them. She could easily imagine. Left to herself, Susan would have warned the removers about the piano before accepting the estimate. Robert had said sternly that it was their business to see the problem, and their bad luck if they didnt. The piano now stood in solitary glory in the upstairs sitting room, the best room in the house. They would leave it till last, naturally. Sitting on the bottom stair, for all the chairs were gone now, she remembered the time they had arrived. The day she was living through now was like that day filmed and run backwards - the piano had been carried in first. And it had got stuck on the stairs. For nearly two hours the team of removal men struggled manfully with it, until it seemed they would simply have to give up. They brought it up to the turn of the stairs, and down again, and cut out banister rails, and got it jammed anyway, while little Robert looked on enthralled, and young Francesca wailed, We cant live in a house without a piano! We cant! Id rather die! And of course they couldnt, not with a musical daughter destined to be a concert pianist. They had to find a way to get it in, and a way had been found. Peter, her late husband, had come home to the crisis and had resolved it. The piano had been left in the garden while the other furniture was brought in - there was much less of it then, they had been relatively young and hard up. And next day, to everyones surprise, a builder had been engaged to take out the first floor window. Then, from the quay below the house, where fish were unloaded from the inshore boats, a little crane was borrowed, and dragged up the hill by means of the local farmers tractor. Finally, the piano was wrapped in blankets, hooked to the crane and gently swung safely through the gaping window, while the entranced children danced with joy at the sight of it. However, the whole process had cost so much it was months before they could afford to have the piano professionally tuned. Thats that, Peter had said. Thats there for ever. But for ever is a long time. The children were increasingly too busy to come home at weekends, and Susan was no longer so mobile in the house, and puffing as she climbed the stairs. The thought of the stairs interrupted her daydream. The banister rails were still not quite parallel, they had not been put back perfectly all that time ago. She ought to have warned the removers, surely she ought. But now it was too late. Any moment now they would find it. She looked around, dazed and panic-stricken. Are you all right, love? Fred was saying. Mind yourself, its just the piano to come now, and then well be on our way. She moved from the bottom stair, heart beating. Robert and Francesca had both appeared, standing in the back of the hallway to watch. No tears then?' Fred said, conversationally. Truth to tell she was just on the edge of them. How odd that simply moving things made them matter. Chairs and cups and things, hundreds of things, that one never noticed or gave a moments thought to while they stayed put, now they were displaced, were full of pathos, crying out to be cared about - and she would have cried, in a moment, surely she would. Only just then the piano appeared, lurching at the top of the stairs, with Fred backing down in front of it and one of the others behind. It tipped slightly. Easy does it! cried Fred, and they carried smoothly down the stairs and out of the front door, and put it down behind the removal van on the road. It was Robert who laughed first, but then they couldn't stop laughing, relieved that it was all over. All three of them, helplessly, leaning against each other, gasping for breath and laughing more. Whats the joke, then? asked Fred, but he merely started them off again. So that, as they went, the three of them, arm in arm down the path for the last time, the only tears she shed were tears of laughter.

- Who would have warned the removers about the piano before accepting the? *Ungrammatical
- Who observed Susan does not look like she's going to cry on the years?
- What thought of the stairs interrupted Susan's daydream? *Vague

The first question is ungrammatical and leaves out the ending noun phrase. This issue is likely due to an incorrect parsing of the original source sentence. The third question is also vague and awkwardly phrased.

Bibliography

Giuseppe Attardi, Stefano Dei Rossi, Giulia Di Pietro, Alessandro Lenci, Simonetta Montemagni, and Maria Simi. A resource and tool for supersense tagging of italian texts. In *LREC*, 2010.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

Zhiqiang Cai, Vasile Rus, Hyun-Jeong Joyce Kim, Suresh C. Susarla, Pavan Karnam, and Arthur C. Graesser. Nlgml: A markup language for question generation. In Thomas Reeves and Shirley Yamashita, editors, *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2006*, pages 2747–2752, Honolulu, Hawaii, USA, October 2006. Association for the Advancement of Computing in Education (AACE). URL `https://www.learntechlib.org/p/24121`.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio.

Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics, 2006.

Massimiliano Ciaramita and Mark Johnson. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 168–175. Association for Computational Linguistics, 2003.

Michael Collins. Probabilistic context-free grammars (pcfgs). *Lecture Notes*, 2013.

Ronan Collobert and Samy Bengio. Links between perceptrons, mlps and svms. In *Proceedings of the twenty-first international conference on Machine learning*, page 23. ACM, 2004.

Deborah Coughlin. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT summit IX*, pages 63–70, 2003.

Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*, 2017.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*, 2016.

Michael Heilman. *Automatic factual question generation from text.* PhD thesis, Carnegie Mellon University, 2011.

Michael Heilman and Noah A Smith. Question generation via overgenerating transformations and ranking. Technical report, DTIC Document, 2009.

Michael Heilman and Noah A Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics, 2010.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. URL `http://arxiv.org/abs/1506.03340`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

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

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Roger Levy and Galen Andrew. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 2231–2234, 2006.

David Lennart Lindberg. *Automatic question generation from text for self-directed learning.* PhD thesis, Applied Sciences: School of Computing Science, 2013.

Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*, 2014.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

Ani Nenkova, Kathleen McKeown, et al. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233, 2011.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Yohei Seki. Sentence extraction by tf/idf and position weighting from newspaper articles. 2002.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics, 2012.

Menglin Xia, Ekaterina Kochmar, and E Briscoe. Text readability assessment for second language learners. 2016.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*, 2017.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792*, 2017.