Answering Definition Questions: Dealing with Data Sparseness in Lexicalised Dependency Trees-based Language Models

Alejandro Figueroa[1] and John Atkinson[2]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz - DFKI,
Stuhlsatzenhausweg 3, D - 66123, Saarbrücken, Germany
figueroa@dfki.de,
WWW home page: http://www.dfki.de/~figueroa
[2] Department of Computer Sciences,
Universidad de Concepción, Concepción, Chile
atkinson@inf.udec.cl,
WWW home page: http://www.inf.udec.cl/~atkinson

Abstract. A crucial step in the answering process of definition questions, such as "*Who is Gordon Brown?*", is the ranking of answer candidates. In definition Question Answering (QA), sentences are normally interpreted as potential answers, and one of the most promising ranking strategies predicates upon Language Models (LMs).
However, one of the factors that makes LMs less attractive is the fact that they can suffer from data sparseness, when the training material is insufficient or candidate sentences are too long. This paper analyses two methods, different in nature, for tackling data sparseness head-on: (1) combining LMs learnt from different, but overlapping, training corpora, and (2) selective substitutions grounded upon part-of-speech (POS) taggings.
Results show that the first method improves the Mean Average Precision (MAP) of the top-ranked answers, while at the same time, it diminishes the average F-score of the final output. Conversely, the impact of the second approach depends on the test corpus.

Key words: Web Question Answering, Definition Questions, Lexical Dependency Paths, Definition Search, Definition Question Answering, n-gram Language Models, Data Sparseness

1 Introduction

Answering definition questions differ markedly from traditional factoid questions. Factoid questions require a single fact to be returned to the user. Conversely, definition questions consist in a substantially more complex response, which should describe the most relevant aspects of the topic of the question (aka. *definiendum* or target).

By and large, typical definitional QA systems rank candidate sentences taken from multiple documents, select the top-ranked candidates, and consolidates

them into a final output afterwards. Broadly speaking, answering a definition query involves a zooming process that comprises the following steps: search, document processing, sentence ranking, summarisation, and many times sense discrimination.

In essence, definition QA systems focus on discovering answers to definition questions by gathering a succinct, diverse and accurate set of factual pieces of information about the *definiendum*. In the QA jargon, these pieces of information are usually called *nuggets*. The following sentence, for instance, yields an answer to the query "*Who is Gordon Brown?*":

```
Gordon Brown is a British politician and leader of the Labour
Party.
```

This illustrative phrase provides the next three nuggets:

```
British
politician
leader of the Labour Party
```

Specifically, answers to questions asking about politicians can encompass important dates in their lives (birth, marriage and death), their major achievements and any other interesting items such as party membership or leadership. As in our working example, a sentence can certainly carry several nuggets.

Our previous work [8] investigated the extent to which descriptive sentences, discovered on the web, can be characterised by some regularities in their lexicalised dependency paths. In particular, in sentences that match some definition patterns such as "*is a*", "*was the*" and "*became the*". For example, the next two paths can be taken from our working example:

`ROOT→is→politician`
`politician→leader→of→Entity`

The first path acts as a *context indicator* (*politician*) signalling the type of *definiendum* being described, whereas the latter yields content that is very likely to be found across descriptions of several instances of this particular context indicator. One interesting facet of [8] is that they make an inference process that clusters sentences according to a set of context indicators (e.g., song, disease, author) found across Wikipedia articles, and built an n-gram LM for each particular context afterwards. Test sentences are ranked thereafter according to its respective language model. As a result, we found out that regularities in dependency paths proved to be salient indicators of definitions within web documents.

In this work, we extend the research presented in [8] by studying two different ways of tackling data sparseness. The first aims at extracting contextual language models from different snapshots of Wikipedia, and the second is aimed specifically at using linguistic abstractions of some pre-determined syntactic classes. The latter strategy shares the same spirit with [5, 6].

The organisation of this paper is as follows: section 2 discusses the related approach to definitional question answering, section 3 describes the acquisition process of our training material, our language models and answer extraction

strategy, section 4 shows the results obtained by applying our approach and finally section 5 highlights the main conclusions and further work.

2 Related Work

Definition QA systems are usually assessed as a part of the QA track of the *Text REtrieval Conference* (TREC). Definition QA systems attempt to extract answers from a target collection of news documents: the AQUAINT corpus. In order to discover correct answers to definition questions, definition QA systems extract nuggets from several external specific resources of descriptive information (e.g. online encyclopedia and dictionaries), and must then project them into the corpus afterwards. Generally speaking, this projection strategy relies on two main tasks:

1. Extract external resources containing entries corresponding to the *definiendum*.
2. Find overlaps between terms in definitions (within the target collection) and terms in the specific resources.

In order to extract sentences related to the *definiendum*, some approaches take advantage of external resources (e.g., *WordNet*), online specific resources (e.g., Wikipedia) and Web snippets [7]. These are then used to learn frequencies of words that correlate with the *definiendum*. Experiments showed that definitional websites greatly improved the performance by leaving few unanswered questions: Wikipedia covered 34 out of the 50 TREC–2003 definition queries, whereas *biography.com* did it with 23 out of 30 questions regarding people, all together providing answers to 42 queries. These correlated words were then used to form a centroid vector so that sentences can be ranked according to the cosine distance to this vector. The peformance of this class of strategies, however, fell into a steep decline when the *definiendum* cannot be found in knowledge bases [11, 20].

One advantage of this kind of approach is that this ranks candidate answers according to the degree in which their respective words characterise the *definiendum*, which is the principle known as the Distributional Hypothesis[10, 12]. However, the approach fails to capture sentences containing the correct answers with words having low correlation with the *definiendum*. This in turn causes a less diverse output and so decrease the coverage. In addition, taking into account only semantic relationships is not sufficient for ranking answer candidates: the co-occurrence of the *definiendum* with learnt words across candidate sentences does not necessarily guarantee that they are syntactically dependent. An example of this can be seen in the following sentence about "*British Prime Minister Gordon Brown*":

`According to the Iraqi` **Prime Minister**`'s office, Gordon Brown was reluctant to signal the withdrawal of` **British** `troops.`

In order to deal with this issue, [4] introduced a method that extended centroid vectors to include word dependencies which are learnt from the 350 most

frequent stemmed co-occurring terms taken from the best 500 snippets retrieved by *Google*. These snippets were fetched by expanding the original query by a set of highly co-occurring terms. These terms co-occur with the *definiendum* in sentences obtained by submitting the original query plus some task specific clues, (e.g.,"*biography*"). Nevertheless, having a threshold of 350 frequent words is more suitable for technical or accurate *definiendums* (e.g., "*SchadenFreude*"), than for ambiguous or biographical *definiendums* (e.g., "*Alexander Hamilton*") which need more words to describe many writings of their several facets. These 350 words are then used for building an ordered centroid vector by retaining their original order within the sentences. To illustrate this, consider the following example[4]:

```
Today's Highlight in History: On November 14, 1900, Aaron Copland,
one of America's leading 20th century composers, was born in
New York City.
```

The corresponding ordered centroid vectors become the words:

```
November 14 1900 Aaron Copland America composer born
New York City.
```

These words are then used for training statistical LMs and ranking candidate answers. Bi-term language models were observed to significantly improve the quality of the extracted answers showing that the flexibility and relative position of lexical terms capture shallow information about their syntactic relation [1].

Our previous work [8] ranks answer candidates according to n-grams (n=5) LMs built on top of our contextual models, contrary to the trend of definition QA systems that solely utilise articles in knowledge bases corresponding to the *definiendum*. Our context models assist in reducing the narrow coverage provided by knowledge bases for many *definiendums*. These n-grams models are learnt from sequences of consecutive lexicalised terms linked in dependency trees representing the sentences in each context.

The contribution of this work is an analysis of two different strategies for reducing data sparseness, when using dependency path-based LMs, and thus enhancing the ranking strategy proposed in our previous work[8].

3 Our Approach

In the following section, the three main parts of our method introduced in [8] are pined down: definition sentence clustering, learning language models and ranking answer candidates.

3.1 Grouping Sentences According to their Contexts Indicators

In our approach, *context indicators* and their corresponding dependency paths are learnt from abstracts extracted from Wikipedia[1]. Specifically, contextual n-gram language models are constructed on top of these contextual dependency

[1] We used the snapshot supplied in January 2008.

Author
CONCEPT was a Entity author of children's books .
CONCEPT was a Entity author and illustrator of children's books .
CONCEPT is the author of two novels : Entity and Entity .
CONCEPT is an accomplished author .
CONCEPT is an Entity science fiction author and fantasy author .
CONCEPT is a contemporary Entity author of detective fiction .
Player
CONCEPT is a Entity football player , who plays as a midfielder for Entity .
CONCEPT is a Entity former ice hockey player .
CONCEPT is a Entity jazz trumpet player .
Disease
CONCEPT is a fungal disease that affects a wide range of plants .
CONCEPT is a disease of plants , mostly trees , caused by fungi .
CONCEPT is a chronic progressive disease for which there is no cure .
Song
CONCEPT is a Entity song by Entity taken from the Entity album Entity .
CONCEPT is a Entity song performed by the Entity band Entity .
CONCEPT is a pop song written by Entity and Entity , produced by Entity for Entity's first album Entity .

Table 1. Some examples of grouped sentences according to their context indicators

paths in order to recognise sentences conveying definitions. Unlike other QA systems [13], definition patterns are applied at the surface level [17] and key named entities are identified using named-entity recognisers (NER)[2]. Preprocessed sentences are then parsed by a lexicalised dependency parser[3], in which obtained lexical trees are used for building a treebank of lexicalised definition sentences.

The treebank contains trees for $1,900,642$ different sentences in which each entity is replaced with a placeholder. This placeholder allows us to reduce the sparseness of the data and to obtain more reliable frequency counts. For the same reason, we left unconsidered different categories of entities and capitalised adjectives were mapped to the same placeholder.

From the sentences in the treebank, our method automatically identifies potential *Context Indicators*. These involve words that signal what it is being defined or what type of descriptive information is being expressed. Context indicators are recognised by walking through the dependency tree starting from the root node. Since only sentences matching definition patterns are taken into account, there are some regularities that are useful to find the respective context indicator. The root node itself is sometimes a context indicator, however, whenever the root node is a word contained in the surface patterns (e.g. *is*, *was* and *are*), the method walks down the hierarchy. In the case that the

[2] http://nlp.stanford.edu/software/CRF-NER.shtml
[3] http://nlp.stanford.edu/software/lex-parser.shtml

root has several children, the first child is interpreted as the context indicator. Note that the method must sometimes goes down one more level in the tree depending of the expression holding the relationship between nodes (i.e., "*part/kind/sort/type/class/first of*"). Furthermore, the used lexical parser outputs trees that meet the projection constrain, hence the order of the sentence is preserved. Overall, 45,698 different context indicators were obtained during parsing.

Candidate sentences are grouped according to the obtained context indicators (see table 1). Consequently, highly-frequent directed dependency paths within a particular context are hypothesised to significantly characterise the meaning when describing an instance of the corresponding context indicator. This is strongly based on the extended distributional hypothesis [14] which states that if two paths tend to occur in similar contexts, their meanings tend to be similar. In addition, the relationship between two entities in a sentence is almost exclusively concentrated in the shortest path between the two entities of the undirected version of the dependency graph [2]. Hence one entity can be interpreted as the *definiendum*, and the other can be any entity within the sentence. Therefore, paths linking a particular type of *definiendum* with a class of entity relevant to its type will be highly frequent in the context (e. g., `politician`→ `leader`→ `of` → `ENTITY`).

Enriching our Treebank with POS Information This treebank is built on top of our previous one, but it accounts for selective substitutions. Contrary to [5, 6], the following syntactic categories are mapped into a placeholder indicating the respective class: DT, CC, PRP, PRP$,CD, RB, FW, MD, PDT, PRP, RBR, RBS and SYM. Additionally, the following verbs, which are normally used for discovering definitions, are mapped into a placeholder: *is*, *are*, *was*, *were*, *become*, *becomes*, *became*, *had*, *has* and *have*. The aim of these mappings is amalgamating the probability mass of similar paths, when computing our language models. For example, the following illustrative paths:

was→politician→a
is→politician→the
is→politician→an

These paths are merged into: VERB→politician→DT. The idea behind this amalgamation is supported by the fact that some descriptive phrases, including "*Concept was an American politician...*" and "*Concept is a British politician...*", share some common structure that is very likely to convey definitions. Consolidating thus their probability mass is reasonable, because it boosts the chances of paths not seen in the training data, but that share some syntactic structures.

3.2 Building Contextual Language Models

For each context, all directed paths containing two to five nodes are extracted. Longer paths are not taken into consideration as they are likely to indicate

weaker syntactic/semantic relations. Directions are mainly considered because relevant syntactical information regarding word order is missed when going up the dependency tree. Otherwise, undirected graphs would lead to a significant increase in the number of paths as it might go from any node to any other node. Some illustrative directed paths obtained from the treebank for the context indicator *politician* are shown below:

politician→affiliated→with→Entity
politician→considered→ally→of→Entity
politician→during→time→the
politician→head→of→state→of
politician→leader→of→opposition
politician→member→of→chamber
politician→served→during
proclaimed→on→Entity

From the obtained dependency paths, an n-gram statistical language model ($n = 5$) for each context was built in order to estimate the most relevant dependency path. The probability of a dependency path **dp** in a context c_s defines the likely dependency links that compose the path in the context c_s, with each link probability conditional on the last $n-1$ linked words:

$$P(\mathbf{dp} \mid c_s) \approx \prod_{i=1}^{l} P(w_i \mid c_s, w_{i-n+1}^{i-1}) \tag{1}$$

Where $P(w_i \mid c_s, w_{i-n+1}^{i-1})$ is the probability that word w_i is linked with the previous word w_{i-1} after seeing the dependency path $w_{i-n+1} \ldots w_{i-1}$. In simple words, the likelihood that w_i is a dependent node of w_{i-1}, and w_{i-2} is the head of w_{i-1}, and so forth.

The probabilities $P(w_i \mid c_s, w_{i-n+1}^{i-1})$ are usually computed by computing the *Maximum Likelihood Estimate*:

$$P_{ML}(w_i \mid c_s, w_{i-n+1}^{i-1}) = \frac{count(c_s, w_{i-n+1}^{i})}{count(c_s, w_{i-n+1}^{i-1})}$$

Some illustrative examples are as follows:

$$P_{ML}(Entity \mid politician, politician \rightarrow affiliated \rightarrow with) =$$

$$\frac{count(politician, politician \rightarrow affiliated \rightarrow with \rightarrow Entity)}{count(politician, politician \rightarrow affiliated \rightarrow with)} = 0.875$$

$$P_{ML}(of \mid politician, politician \rightarrow activist \rightarrow leader) =$$

$$\frac{count(politician, politician \rightarrow activist \rightarrow leader \rightarrow of)}{count(politician, politician \rightarrow activist \rightarrow leader)} = 0.1667$$

$$P_{ML}(Entity \mid politician, proclaimed \rightarrow on) =$$

$$\frac{count(politician, proclaimed \rightarrow on \rightarrow Entity)}{count(politician, proclaimed \rightarrow on)} = 1$$

However, in our case, the word count $c(c_s, w_{i-n+1}^{i})$ can frequently be greater than $c(c_s, w_{i-n+1}^{i-1})$. For example, in the following definition sentence:

`CONCEPT is a band formed in Entity in Entity.`

The word "*formed*" is the head of two "*in*", hence the denominator of $P(w_i \mid c_s, w_{i-n+1}^{i-1})$ is the number of times w_{i-1} is the head of a word (after seeing w_{i-n+1}^{i-1}).

In order to illustrate how selective substitutions assist in consolidating the probability mass according to some syntactic similarities at the category level, consider the next example:

$$\begin{array}{ll} P_{ML}(a \mid is \rightarrow politician) & = 0.164557 \\ P_{ML}(an \mid is \rightarrow politician) & = 0.0379747 \\ \underline{P_{ML}(the \mid is \rightarrow politician)} & \underline{= 0.00632911\ +} \\ P_{ML}(DT \mid is \rightarrow politician) & = 0.20886081 \end{array}$$

The obtained 5-gram language model is smoothed by interpolating with shorter dependency paths as follows:

$$P_{interp}(w_i \mid c_s, w_{i-n+1}^{i-1}) = \lambda_{c_s, w_{i-n+1}^{i-1}} P(w_i \mid c_s, w_{i-n+1}^{i-1}) + (1 - \lambda_{c_s, w_{i-n+1}^{i-1}}) P_{interp}(w_i \mid c_s, w_{i-n+2}^{i-1})$$

The probability of a path $P(\mathbf{dp} \mid c_s)$ is accordingly computed by accounting for the recursive interpolated probabilities instead of raw Ps. Note also that $\lambda_{c_s, w_{i-n+1}^{i-1}}$ is computed for each context c_s as described in [3]. Finally, a sentence S is ranked according to its likelihood of being a definition as follows:

$$rank(S) = P(c_s) \sum_{\forall \mathbf{dp} \in S} P(\mathbf{dp} \mid c_s) \tag{2}$$

In order to avoid counting redundant dependency paths, only paths ending with a **dependent/leave node** are taken into account, whereas **duplicate** paths are discarded.

Combining Context Models from Different Wikipedia Snapshots Another way of tackling data sparseness is amalgamating LMs learnt from different Wikipedia[4] snapshots. Following the same procedure described in section 3, two additional treebanks of dependency trees were built, and hence two extra n-gram language models were generated. Accordingly, the ranking of a candidate sentence S was computed by making allowances for average values of $P(c_s)$ and $P(\mathbf{dp} \mid c_s)$.

[4] For this purpose, we took advantage of two additional snapshots. One corresponding to early 2007 and the other to October 2008. The former yielded 1,549,615 different descriptive sentences, whereas the latter 1,063,452.

$$\bar{rank}(S) = \frac{1}{B}\sum_{b=1}^{B} P_b(c_s) * \sum_{\forall \mathbf{dp} \in S} \frac{1}{B}\sum_{b=1}^{B} P_b(\mathbf{dp} \mid c_s) \quad (3)$$

In other words, we carry out experiments by systematically increasing the size of our language models in three steps $B = 1, 2, 3$. In the previous equation, $P_b(c_s)$ is the probability of the context c_s in the treebank b, and by the same token, $P_b(\mathbf{dp} \mid c_s)$ is the probability of finding the dependency path $\mathbf{dp}$ in the context c_s in the treebank b. Accordingly, $\bar{rank}(S)$ is the final ranking value, and when $B = 1$, $\bar{rank}(S)$ is equal to $rank(S)$, which resembles our original system presented in [8].

3.3 Extracting Candidate Answers

Our model extracts answers to definition questions from web snippets. Thus, sentences matching definition patterns at the surface level are pre-processed[5] and parsed in order to get the corresponding lexicalised dependency trees. Given a set of test sentences/dependency trees extracted from the snippets, our approach discovers answers to definition question by iteratively selecting sentences.

```
1  φ = ∅;
2  indHis = getContextIndicatorsHistogram(T);
3  for highest to lowest frequent ι ∈ indHis do
4      while true do
5          nextSS = null;
6          forall t_t ∈ T do
7              if ind(t_i)==ι then
8                  rank = rank(t_i,φ);
9                  if nextSS == null or rank > rank(nextSS) then
10                     nextSS = t_i;
11                 end
12             end
13         end
14         if nextSS == null or rank(nextSS) ≤ 0.005) then
15             break;
16         end
17         print nextSS;
18         addPaths(nextSS,φ);
19     end
20 end
```

Algorithm 1: Answer Extractor

The general strategy for this iterative selection task can be seen in algorithm 1 whose input is the set of dependency paths (T). This first initialises a set ϕ

[5] http://www.comp.nus.edu.sg/~qiul/NLPTools/JavaRAP.html

which keeps the dependency paths belonging to previously selected sentences (line 1). Next, context indicators for each candidate sentence are extracted so as to build an histogram $indHist$ (line 2). Since highly frequent context indicators show more reliable potential senses, the method favours candidate sentences based on their context indicator frequencies (line 3). Sentences matching the current context indicator are ranked according to equation 3 (lines 7 and 8). However, only paths **dp** in $t_i - \phi$ are taken into consideration, while computing equation 3. Sentences are thus ranked according to their novel paths respecting to previously selected sentences, while at the same time, sentences carrying redundant information decrease their ranking value systematically. Highest ranked sentences are selected after each iteration (line 9-11), and their corresponding dependency paths are added to ϕ (line 18). If the highest ranked sentence meets the halting conditions, the extraction task finishes. Halting conditions ensure that no more sentences to be selected are left and no more candidate sentences containing novel descriptive information are chosen.

In this answer extraction approach, candidate sentences become less relevant as long as their overlap with all previously selected sentences become larger. Unlike other approaches which control the overlap at the word level [4, 13], our basic unit is a dependency path, that is, a group of related words. Thus, the method favours novel content, while at the same time, it makes a global check of the redundant content. Furthermore, the use of paths instead of words as unit ensures that different instances of a word, that contribute with different descriptive content, will be accounted accordingly.

4 Experiments and Results

In order to assess our initial hypothesis, a prototype of our model was built and assessed by using 189 definition questions taken from TREC 2003-2004-2005 tracks. Since our model extracts answers from the web, these TREC dataset were only used as a reference for question sets. For each question, the best 300 web snippets were retrieved by using MSN Search along with the search strategy sketched in [9]. These snippets were manually inspected in order to create a gold standard. It is important to note that there was no descriptive information for 11 questions corresponding to the TREC 2005 data set. For experiment purposes, we utilised Our System presented in [8] as a baseline, and all systems were provided with the same set of snippets.

4.1 Evaluation Metrics

In this work, two different metrics were allowed: F-score and MAP. Following the current trends of assessments in definition QA systems, the standard F-score [19] was used as follows:

$$\mathcal{F}_\beta = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R}$$

This takes advantage of a factor β for balancing the length of the output and the amount of relevant and diverse information it carries. In early TREC tracks, β was set to 5, but as it was inclined to favour long responses, it was later decreased to 3. The Precision (P) and the Recall (R) metrics were computed as described in the most recent evaluation by using uniform weights for the nuggets [15] in the gold standard obtained in the previous section.

One of the disadvantages of the F-score is that it does not account for the order of the nuggets within the output. This is a key issue whenever definition QA systems output sentences as it is also necessary to assess the ranking order, that is, determine whether the highest positions of the ranking contain descriptive information. In order to deal with this, the *Mean Average Precision* (MAP) was accounted for. Despite an important number of MAP [16], those measuring the precision at fixed low levels of results were used, in particular, MAP-1 and MAP-5 sentences. Hence this precision is referred to as *precision at k*:

$$MAP(Q) = \frac{1}{\mid Q \mid} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision\_at\_k$$

Here Q is a question set (e.g., TREC 2003), and m_j is the number of ranking sentences in the output. Accordingly, m_j is truncated to one or five, when computing MAP-1 and MAP-5, respectively. This metric was selected because its ability to show how good the results are on the first positions of the ranking. Simply put, for a given question set Q, MAP-1 shows the fraction of questions that ranked a valid definition on the top.

4.2 Experimental Results

Table 2 highlights the obtained results. In this table, Our System II ($B = 2$) and Our System III ($B = 3$) correspond to our systems accounting for two and three treebanks, respectively. Overall, the performance was decreased in terms of recall and precision. The gradual decrease in recall may be due to the fact that averaging the two/three treebanks diminishes the value of low frequent paths, because they are not (significantly) present in all the treebanks. Therefore, whenever they match a sentence, the sentence is less likely to score high enough to surpass the experimental threshold (line 14 in algorithm 1). Here, we envisage using a strategy of inter-treebank smoothing that takes away probability mass of the high frequent paths (across treebanks) and distribute it across paths low in frequency in one of the treebanks, but absent in one of the others. The reason to the steady decrease in precision is two-fold:

- The decrease in recall brings about a decrease in the allowance,
- And more important, the algorithm selected misleading or redundant definitions in replacement for the definitions matched by the original system, but missed by these two extensions.

This outcome homologates the fact that ranking answer candidates, according to some highly frequent words across articles about the *definiendum* taken

	TREC 2003	TREC 2004	TREC 2005
Size	50	64	(64)/75
	Our System		
Recall	**0.57**	**0.50**	0.42
Precision	**0.39**	**0.40**	**0.29**
F(3) Score	**0.53**	**0.47**	0.38
	Our System II		
Recall	0.46	0.46	0.42
Precision	0.32	0.38	**0.29**
F(3) Score	0.43	0.44	0.38
	Our System III		
Recall	0.46	0.44	0.41
Precision	0.31	0.34	0.28
F(3) Score	0.43	0.42	0.37
	Our System POS		
Recall	0.56	0.47	**0.48**
Precision	0.24	0.22	0.24
F(3) Score	0.48	0.41	**0.42**

Table 2. Results for TREC question sets.

from several knowledge bases, would bring about an improvement in terms of ranking, but a detriment to coverage and to the diversity of the final output.

On the other hand, highly frequent paths obtain more robust estimates as they are very likely to be in all treebanks, having a positive effect in the ranking. Table 3 highlights this effect. In all question sets, Our System II and Our System III outperformed our original system. The increase in MAP values suggests that combining estimates from different snapshots of Wikipedia assists in determining more prominent and genuine paths. These estimates along with the preference given by algorithm 1 to these paths bring about the improvement in the final ranking, that is more genuine pieces of descriptive information tend to be conveyed in the highest positions of the rank.

In general, our three improvements bettered the ranking with respect to Our System, however our experiments did not draw a clear distinction which is the best in this aspect. For our POS based method, results in table 3 indicates an increase with respect to the original system for two datasets, but a decrease in the case of the TREC 2005 questions set. Unlike the two previous question sets, abstracting some syntactic categories leaded to some spurious sentences to rank higher.

More interestingly, table 2 emphasises the marked decline in terms of F(3)-score for two datasets, while at the same time, it remarks a substantial improvement for the TREC 2005 question set. In comparison with the results achieved by the original system sketched in table 2. This enhancement is particularly due to the increase in recall so that the amalgamation of dependency paths was useful to identify a higher number of genuine descriptive sentences. On the other hand, the addition of POS tags assisted in matching more misleading and

	Our System	Our System II	Our System III	Our System POS
		TREC 2003		
MAP-1	0.82	**0.88**	**0.88**	**0.88**
MAP-5	0.82	**0.88**	0.87	**0.88**
		TREC 2004		
MAP-1	0.88	0.92	**0.94**	0.91
MAP-5	0.82	**0.88**	0.87	0.87
		TREC 2005		
MAP-1	0.79	0.81	**0.82**	0.73
MAP-5	0.77	**0.78**	**0.78**	0.71

Table 3. *Mean Average Precision* (MAP).

spurious sentences, and consequently it worsened the performance in terms of precision. This might also explain the decrease in the MAP value for this question set. Given these observations, our treebanks (without POS information) were observed to cover less descriptive sentences contained in this question set. In the TREC 2003-2004 question sets, the decline might be due to the fact that different original paths are still necessary to recognise several sentences.

5 Conclusions

In this work, we studied two different approaches to tackle data sparseness, when utilising n-grams language models built on top of dependency paths for ranking definition questions.

Results show that the precision of the top-ranked answers can be boosted by combining contextual language models learnt from different snapshots of Wikipedia. However, this can have a negative impact in the precision and the diversity of the entire output. Additionally, our experiments showed that the success of abstractions based on POS taggings depends largely upon the target corpus. Nevertheless, a study of the effects of additional features in our languages models can be done as a further work. A study similar in spirit to [18].

Acknowledgments

This work was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project `HyLaP` (FKZ: 01 IW F02) and the EC- funded project QALL-ME - FP6 IST-033860 (http://qallme.fbk.eu). Additionally, this research was partially sponsored by the National Council for Scientific and Technological Research (FONDECYT, Chile) under grant number 1070714.

References

1. M. Belkin, J. Goldsmith, Using eigenvectors of the bigram graph to infer grammatical features and categories, in: Proceedings of the Morphology/Phonology Learning Workshop of ACL-02, 2002.
2. R. Bunescu, R. J. Mooney, A Shortest Path Dependency Kernel for Relation Extraction, in: Proceedings of HLT/EMNLP, 2005.
3. S. Chen, J. Goodman, An Empirical Study of Smoothing Techniques for Language Modeling, in: Proceedings of the 34th Annual Meeting of the ACL, 1996.
4. Y. Chen, M. Zhon, S. Wang, Reranking Answers for Definitional QA Using Language Modeling, in: Coling/ACL-2006, 2006.
5. H. Cui, M.-Y. Kan, T.-S. Chua, Unsupervised Learning of Soft Patterns for Definitional Question Answering, in: Proceedings of the Thirteenth World Wide Web Conference (WWW 2004), 2004.
6. H. Cui, M.-Y. Kan, T.-S. Chua, Soft pattern matching models for definitional question answering, ACM Trans. Inf. Syst. 25 (2).
7. T. Cui, M. Kan, J. Xiao, A comparative study on sentence retrieval for definitional question answering, in: SIGIR Workshop on Information Retrieval for Question Answering (IR4QA), 2004.
8. A. Figueroa, J. Atkinson, Using Dependency Paths For Answering Definition Questions on The Web, in: 5th International Conference on Web Information Systems and Technologies, 2009.
9. A. Figueroa, G. Neumann, A Multilingual Framework for Searching Definitions on Web Snippets, in: KI, 2007.
10. J. R. Firth, A synopsis of linguistic theory 1930-1955, in: Studies in Linguistic Analysis, 1957.
11. K. Han, Y. Song, H. Rim., Probabilistic model for definitional question answering, in: Proceedings of SIGIR 2006, 2006.
12. Z. Harris, Distributional Structure, in: Distributional structure. Word, 10(23), 1954.
13. W. Hildebrandt, B. Katz, J. Lin, Answering Definition Questions Using Multiple Knowledge Sources, in: Proceedings of HLT-NAACL, 2004.
14. D. Lin, P. Pantel, Discovery of Inference Rules for Question Answering, in: Journal of Natural Language Engineering, Volume 7, 2001.
15. J. Lin, D. Demner-Fushman, Will pyramids built of nuggets topple over?, in: Proceedings of the main conference on HTL/NAACL, 2006.
16. C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
17. M. M. Soubbotin, Patterns of Potential Answer Expressions as Clues to the Right Answers, in: Proceedings of the TREC-10 Conference, 2001.
18. M. Surdeanu, M. Ciaramita, H. Zaragoza, Learning to Rank Answers on Large Online QA Collections, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008), 2008.
19. E. M. Voorhees, Evaluating Answers to Definition Questions, in: HLT-NAACL, 2003.
20. Z. Zhang, Y. Zhou, X. Huang, L. Wu, Answering Definition Questions Using Web Knowledge Bases, in: Proceedings of IJCNLP 2005, 2005.