AGRA: An Analysis-Generation-Ranking Framework for Automatic Abbreviation from Paper Titles

Jianbing Zhang, Yixin Sun, Shujian Huang*, Cam-Tu Nguyen,
Xiaoliang Wang, Xinyu Dai, Jiajun Chen, Yang Yu
National Key Laboratory for Novel Software Technology, Nanjing University, China
{zjb,huangsj,waxili,daixinyu,chenjj,yuy}@nju.edu.cn
sunyx@nlp.nju.edu.cn, ncamtu@gmail.com

Abstract

People sometimes choose word-like abbreviations to refer to items with a long description. These abbreviations usually come from the descriptive text of the item and are easy to remember and pronounce, while preserving the key idea of the item. Coming up with a nice abbreviation is not an easy job, even for human. Previous assistant naming systems compose names by applying hand-written rules, which may not perform well. In this paper, we propose to view the naming task as an artificial intelligence problem and create a data set in the domain of academic naming. To generate more delicate names, we propose a three-step framework, including description analysis, candidate generation and abbreviation ranking, each of which is parameterized and optimizable. We conduct experiments to compare different settings of our framework with several analysis approaches from different perspectives. Compared to online or baseline systems, our framework could achieve the best results.

1 Introduction

Using abbreviation to name an item with a long description is a common language phenomenon. For example, the abbreviation *IBM* is often used for the International Business Machines Corporation. Abbreviations also play an important role in academic communications. The full name of an approach or a system usually consists of several words summarizing its core concept, which is sometimes difficult to be memorized or referred to. On contrast, word-like abbreviations are more convenient to remember and pronounce, and also remind people of the original names.

Abbreviations are commonly created in different styles (Table 1). For example, *SVM* in example 1 is abbreviated by taking the first letters of all words. In example 2, *AdaBoost* is combined with prefixes extracted from all words with an easy pronunciation. In example 3, *Bagging* is less straightforward, but better for memorization and pronunciation.

Index	Abbreviation	Name of the Items
1	SVM	Support Vector Machine [Cortes and Vapnik, 1995]
2	AdaBoost	Adaptive Boosting [Breiman, 1996]
3	Bagging	Boostrap Aggregating [Breiman, 1996]

Table 1: Examples of abbreviations used in academic circles.

We can see from the examples that giving creative namings, or abbreviations, requires the understanding of the descriptive text, being familiar with the words and pronunciations, etc., which is obviously not an easy job, even for humans. Towards artificial intelligence, it is also interesting to design systems that could perform such a challenging task.

Currently, some abbreviation generation systems have been developed, e.g., `acronymcreator.net`, `naming.com`, `business-name-generators.com`, `netsubstance.com`. These systems use hand-coded rules to generate a list of candidate for human users, which may be helpful when the desired name can be covered by the hand-coded rules. But when the description is long, which leads to a large set of candidates, the hand-coded rules may not be able to help users select better results. We believe that various text understanding techniques, especially for sentences, could be useful in making these suggestions, which is previously little touched [Özbal and Strapparava, 2012].

In this paper, we propose to treat creative naming as an artificial intelligence problem and collect a data set in the domain of academic naming, where abbreviations are generated for an academic paper based on its title.

To solve the problem, we propose a three-step framework: Description Analysis, Candidate Generation and Abbreviation Ranking. To simulate the knowledge that human uses during the naming process, we employ detailed analyses of the descriptive text (paper titles in this case) from different aspects, including lexical, syntactic and semantic levels. The analysis results are then used to guide the generation of candidate abbreviations. These candidates are ranked according to their pronunciation properties. The framework is fully parameterized which provides the ability to be op-

*This work is supported by the National Natural Science Foundation of China (No. 61672277, 71503124, 61370028) and the Collaborative Innovation Center of Novel Software Technology and Industrialization, China. Shujian Huang is the corresponding author. The demonstration system is available at http://nlp.nju.edu.cn/demo/abbreviation.html

Index	Abbreviation	Title Words
1	FAWN	A Fast Array of Wimpy Nodes [Andersen *et al.*, 2008]
2	IPR	An Integrated Placement and Routing Algorithm [Pan and Chu, 2007]
3	SeRQL	A Second Generation RDF Query Language[Broekstra and Kampman, 2003]
4	WBCSim	A Prototype Problem Solving Environment for Wood-Based Composites Simulations [Goel *et al.*, 1998]
5	SIMPLIcity	Semantics-Sensitive Integrated Matching for Picture LIbraries [Wang *et al.*, 2000]

Table 2: Examples of paper titles in the academic naming data set, collected from CiteSeerX.

timized through state-of-the-art derivative-free optimization methods [Martinez-Cantin, 2014; Yu *et al.*, 2016].

Experiments show that hybrid evidences from the syntactic and the semantic relevance analyses can provide the best result, which confirms that these analyses are essential in the name generation process. With the proposed framework, our system generates abbreviations much more similar to those generated by humans, compared to several online systems.

2 The Academic Naming Task

2.1 The Task and Data Set

General purpose creative naming could be used in various ways, e.g. naming a corporation from its business [Özbal and Strapparava, 2012]. In these cases, more sophisticated linguistic tricks, such as analogy or metaphor may be used, which is too wild to handle. Several research focus on the abbreviations of daily Chinese text, which is less creative and could be handled by character level features [Chang and Lai, 2004; Yang *et al.*, 2009].

In this paper, we propose to study a specific domain of creative naming, i.e. academia naming. In academic writings, authors tend to generate an abbreviation for their papers that can represent the meaning of the original title as much as possible and remind people of the content of the paper. We choose to use paper titles for our study, because for most of the papers, the abbreviations of titles are carefully handpicked by authors. And more importantly, in many cases, the abbreviation comes from the words in the title only.

Thus, the aim of task is clear: to generate the abbreviation from the title, which is simpler and more constrained than the general purpose case.

We collect 1000 paper titles with abbreviations from CiterSeerX[1]. Most of these papers are about computer and information science. These titles are split into two parts: the abbreviation part and the title word part (Table 2).

2.2 Manual Analysis of the Abbreviations

We manually examine how these abbreviations are built from titles and list the analysis results in Table 3. About 25% abbreviations are simply the combination of the first characters of all words in the title, without function words such as "the", "a", "of", etc. For example, the abbreviation *FAWN* is generated in the same order from the four words "fast", "array", "wimpy" and "nodes", but not including function words "a" and "of" (example 1 in Table 2).

About 61% papers choose to combine prefixes of certain words in the title (e.g. example 2-4). In example 4, *WBCSim* takes the first character from "wood", "based" and "composites", the three-character prefix from the word "simulations", while taking no character from the first several words, i.e. "prototype", "problem", "solving" and "environment". It takes the three character prefix from "simulations" maybe because that "simulations" is the core concept in the paper.

Abbreviation Type	Num. of Items	Percentage
first letters of all words	247	25%
prefixes of some words	602	61%
word rewriting	78	8%
extra information	53	6%

Table 3: Statistics of abbreviations by types.

The above two types add up to 86% of all abbreviations in our data set, which suggests that most of the abbreviations come from prefixes of title words. The remaining 14% abbreviations are generated in more sophisticated ways, by changing the letters in prefixes extracted from title or using some external words. For example, in example 5 of Table 2, only part of the abbreviation *SIMPLIcity* comes from the title (*SIMPLI*, shown in uppercase). The rest of the abbreviation *city* does not appear in the title. It may come from the content of the paper or from other considerations.

Figure 1 shows the distribution of collected abbreviations according to their length in characters. We can see that most abbreviations are of lengths between 3 to 7 characters (around 95%). Possible reasons are that abbreviations shorter than 3 characters are less informative, while too long abbreviations might be hard to remember.

3 Framework for Automatic Abbreviation

As mentioned earlier, the generation of creative abbreviations is not trivial for human. It requires various ability and knowledge including the understanding of text, handling of word shapes, pronunciation, etc. These challenges make it an in-

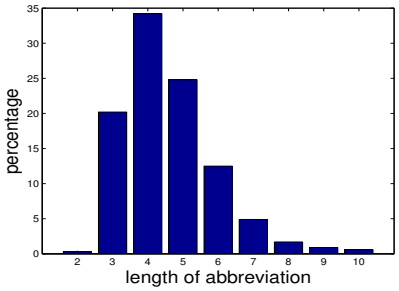


Figure 1: Distribution of abbreviation by length (in characters).

[1]http://citeseerx.ist.psu.edu

teresting question that whether the ability could be learnt by machines. We present in this section our understanding and solution of the problem.

3.1 Intuition

Previous examples show that characters in the abbreviation come from words in the title, for reminding the readers of the content of the title or paper. However, it is not necessary to use all words in the title. Beside functional words, content words may also not be used at all, while longer prefix may be used from some core content words (example 4 in Table 2). This suggests the necessity of treating each word in the title differently, because they have different importance.

Given the preference of words, selecting characters from these words is also a problem. These characters could be a prefix of any length. Selection decisions should be made for each word. And obtaining the whole abbreviation requires an enumeration or search over possible combinations.

Finally, considerations should also be made about the word-shape of the abbreviations, making them easy to remember and pronounce.

Each of the above process requires linguistic knowledge and text understanding. Fortunately, the developing of natural language processing research provides possibilities to carry out the process automatically.

3.2 Architecture

Base on the above intuition, we propose a framework for automatic abbreviation generation, which consists of three steps: Description Analysis, Candidate Generation and Abbreviation Ranking.

With a given descriptive text, our framework firstly go through the step of description analysis to obtain the importance of each word in the title. The second step then employs some strategies to search possible combination of characters and generate candidate abbreviations. The generation of the candidates relies on the word importance derived in the first step; and the patterns that human use for abbreviations should also be concerned. Finally, these candidates will be ranked considering the pronunciation factor, so the abbreviation results will be easy to remember and pronounce.

These steps will be introduced in Section 4- 6, respectively.

3.3 Optimization

In the proposed framework, we model the linguistic knowledge in a feature-based way, where there are tens of weights (i.e., hyper-parameters) to be determined across the three steps. The architecture does not allow computing gradients of these weights throughout the system, and thus they are hard to be optimized. Fortunately, recent development of derivative-free optimization methods, which perform optimization by learning from samples of the weights but need no gradients, has made significant progress both theoretically and practically. Methods with sound theoretical supports and empirical verifications have been available, including the Bayesian optimization [Martinez-Cantin, 2014] and the model-based optimization [Yu *et al.*, 2016; Hu *et al.*, 2017]. In this work, we will employ derivative-free optimization methods to optimize the weights of the system globally.

4 Description Analysis

Analysis of the descriptive text is to determine which words in the text are important and which should not be taken into consideration. In order to distinguish the importance of words, we propose to assign scores for each word in paper title. The higher the score is, the more important the corresponding word is and the more attention should be paid to when constructing abbreviations.

We consider different NLP approaches from lexical, syntactic or semantic levels, respectively.

4.1 Lexical Analysis (Lex)

The generation of abbreviations usually does not consider function words. For convenience, we simply remove all function words in a stop word list from the consideration. We use a relatively small default stop word list, which includes 124 frequently used stop words[2].

Besides stop word, POS tag analysis is another lexical analyzing technique that reveals the function of words in the sentence, which might be an key evidence for importance. We notice that paper titles mainly contain words in 4 categories: verbs, nouns, adjectives and adverbs. So we define 4 indicator features for these categories, respectively.

$$f_{\text{Lex-X}}(w_k) = \begin{cases} 1 & \text{the POS tag of } w_k \text{ is X,} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where X denotes one of the above 4 POS categories.

4.2 Syntactic Analysis (Syn)

Based on lexical analysis, the syntactic analysis examines the grammatical structure of the sentence, which reveals how the sentence is built from words and identifies the relationship between different words.

We notice that the syntactic structure of a sentence also contains evidences for word importance. As shown in Figure 2, the root of the title is the word "environment" which is indeed the object of research in a large scope. However, the phrase "wood-based composites simulations" serves as a modifier to the root word, and states the specific area and aim of the environment, which is actually more important. In fact, the authors picked all the characters of the abbreviation from this modifier phrase.

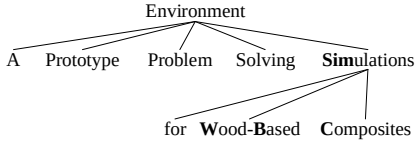


Figure 2: The dependency structure of the title "Prototype Problem Solving Environment for Wood-Based Composites Simulations", with characters for the abbreviation *WBCSim* shown in bold font (Example 4 of Table 2).

[2]http://www.ranks.nl/stopwords/

The above example suggests that modifiers in the title, which are in the bottom layers of the tree, may be more important for describing the content of the title. Thus, we employ a dependency parser to obtain the structural information, and define different indicator features for words in different layers of the tree (i.e. distance to the root node), respectively.

$$f_{\text{Syn-i}}(w_k) = \begin{cases} 1 & w_k \text{ is at the } i^{th} \text{ layer of the tree,} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Generally, most paper titles have a flat and simple structure, so we only use indicator features for the first 3 layers.

4.3 Semantic Analysis (Sem)

We also propose to evaluate the importance of words at the semantic level. Because the meaning of the title is hard to capture, we seek evidences from the semantic relations among title words. Our assumption is that the important word in the title should be semantically related to the other words.

To evaluate the semantic relations between two words, we employ the popular word embeddings [Mikolov *et al.*, 2013]. After training, each word is associated with a vector representation of a fixed length. Relations between words w_i and w_j can be reflected by the cosine distance between vectors, denoted as d_{ij}. We further define a numerical feature as the average cosine distance between w_k and other words in the title:

$$f_{\text{Sem}}(w_k) = \frac{1}{n-1} \sum_{j \neq k} d_{kj} \quad (3)$$

where j enumerates indexes of other words in the title.

4.4 Importance Score for Word

After the previous analysis steps, we define the importance score as the weighted sum of all previous defined features:

$$s_{\text{word}}(w_k) = \vec{w} \cdot [\vec{f}_{\text{Lex}}(w_k), \vec{f}_{\text{Syn}}(w_k), f_{\text{Sem}}(w_k)] \quad (4)$$

where $\vec{f}_{\text{Lex}}$ and $\vec{f}_{\text{Syn}}$ are the vectors of the POS and syntactic features, respectively; $[.,.]$ denotes the concatenation of vectors, and $\vec{w}$ is the corresponding weight vector. Although the features are defined intuitively, the feature weights could be automatically learned from data.

5 Candidate Generation

5.1 Scoring the Candidates

It has been shown that 86% of abbreviations are from the prefixes of title words (Table 3). In the generation steps, we follow this observation and generate the abbreviations by concatenating prefixes of title words.

For convenience, we use $w_1, ..., w_n$ to denote the title words with stop words removed, and p_k to denote the prefix from w_k. So any abbreviation a is a concatenation of prefixes $p_1, ..., p_n$.

We define indicator features for each case where the prefix p_k uses the character at a specific position from the beginning of the word w_k:

$$f_{\text{Char-i}}(p_k, w_k) = \begin{cases} 1 & p_k \text{ uses the } i^{th} \text{ character of } w_k, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The score of the prefix is determined both by the importance of the word, and by the number and positions of characters it fetches from the word:

$$s_{\text{prefix}}(p_k, w_k) = s_{\text{word}}(w_k) \times \vec{w}_{\text{Char}} \cdot \vec{f}_{\text{Char}}(p_k, w_k) \quad (6)$$

where $\vec{f}_{\text{Char}}$ is the vector of character features, $\vec{w}_{\text{Char}}$ is the corresponding weight vector. Thus, by combining these prefixes, we can get the abbreviation candidates. We define the score for candidate a as the sum of the prefix scores averaged by the abbreviation length l:

$$s_{\text{ABBR}}(a) = \frac{1}{l} \sum_{k=1}^{n} s_{\text{prefix}}(p_k, w_k) \quad (7)$$

5.2 Searching for Candidates

With the above scoring functions as guidance, the generation of candidates could be performed as a beam searching process. A priority queue is used to maintain the list of candidates. Each candidate will be marked with its current state, which is a tuple consisting of the current abbreviation a, the last visited word w_k and the length t of the corresponding prefix p_k. The priority is determined by $s_{\text{ABBR}}(a)$. The size of the queue could be limited so the search remains efficient.

At the beginning, the queue is initialized with a single candidate $\langle \emptyset, w_0, 0 \rangle$. At each search step, each item $\langle a, w_k, t \rangle$ in the queue is extended by the following three actions, respectively.

Append Append the next character c of current word to the current prefix, transiting the state to be $\langle a + c, w_k, t + 1 \rangle$

Jump Move to the next word in the title, transiting the state to be $\langle a, w_{k+1}, 0 \rangle$.

Stop Keep the current abbreviation, with no further extension.

The searching process continues until no extension could be made for any item in the beam.

6 Abbreviation Ranking

After the generation step, we obtain a long list of candidates which, at least to some extent, represents the meaning of the title. In this ranking step, we propose methods to pick out those candidates that are easily to pronounce and remember. Özbal and Strapparava [2012] propose to use automatic alignment to map newly generated "words" into their phonetic sequences for modeling their pronunciation properties. But the alignment process usually brings noise.

Instead, we propose a much simple and straight-forward method. We observe that abbreviations that are, or similar to, existing words are usually good candidates, because they are usually easy to pronounce and familiar to people. So instead of modeling the phonetic sequence, we propose to directly model the abbreviations.

To evaluate the similarity between the candidates and existing words, we propose to use the techniques in language modeling [Chen and Goodman, 2010]. In n-gram language models, the probability of a word sequence is modeled as the joint probability of all its words, which is then approximated

with a n^{th} order Markov assumption. Here we adopt n-gram language models to model the abbreviation as a character sequence.

We train the language model on existing words from WordNet [Miller, 1995], with the SRILM toolkit [Stolcke, 2004], and use it to score the candidates. Candidates with higher language model scores are more similar to existing words, thus are considered better for pronunciation.

We denote the language model score of abbreviation a as $f_{\text{LM}}(a)$ and the length of a as $f_{\text{len}}(a)$. Then, we rank all candidates by an adjusted score as follows:

$$s(a) = s_{\text{ABBR}}(a) + w_{\text{LM}} \times f_{\text{LM}}(a) + w_{\text{len}} \times f_{\text{len}}(a) \quad (8)$$

where w_{LM} and w_{len} are the corresponding weights. These parameters could also be optimized given a training data set.

7 Experiments

7.1 Settings

We use 849 titles and abbreviations from the collected data set, with those abbreviations that have the word rewriting and extraneous information (the last two cases in Table 3) removed. Among these items, we randomly select 140 of them for test and use the rest of them as the training set for optimizing the parameters.

To provide analysis for the title words, we use Stanford Parser [Chen and Manning, 2014] to get the POS tags and dependency tree for each title. We use the skip-gram model of the word2vec tool [Mikolov *et al.*, 2013] to train our word embedding on a large paper title data set we collect from `arxiv.org`, containing about 60,000 paper titles. The stop words are removed before training the word embedding.

To evaluate the performance of the system, we borrow the idea of precision@k from information retrieval [Manning *et al.*, 2008] and use a metric called recall@k for our evaluation. Because there is only one single correct abbreviation for each title, the recall@k metric simply reports whether the correct answer is generated in the top k results. In our experiment, we set k as 50.

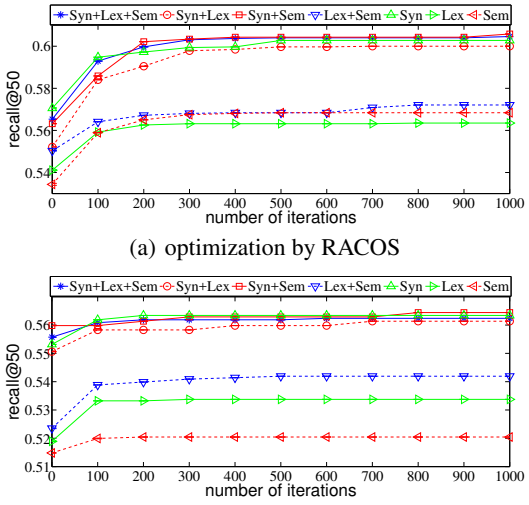
To optimize the parameters for our proposed framework, we apply the state-of-the-art derivative-free optimization techniques, BayesOpt [Martinez-Cantin, 2014] and RACOS[3] [Yu *et al.*, 2016; Hu *et al.*, 2017]. We directly optimize the recall@50 metric, and run these optimization algorithms for 1000 iterations.

7.2 Comparative Experiment

We first experiment with systems that use different combinations of description analysis approaches. The training accuracy curve of optimization with RACOS is shown in Figure 3(a).

We can see that when the three approaches *Syn*, *Lex* and *Sem* are used separately, the performance of *Syn* is much better than *Lex* and *Sem*. It indicates that *Syn* is currently the most crucial factor for finding the keywords in the title, which then leads to better abbreviations.

[3]https://github.com/eyounx/ZOOpt



(a) optimization by RACOS

(b) optimization by BayesOpt

Figure 3: Training performances of systems with different analysis approaches using RACOS and Baysian optimization.

When combined with other approaches, the performance of *Syn+Lex* is worse than *Syn*, which indicates that the use of POS tags may be contradictory with syntactic analysis. The performance of *Lex+Sem* is better than *Lex* and *Sem* alone, shows that the two approaches could complement each other. Among all systems, *Syn+Sem* shows the best training performance, which we will use for further comparisons.

The training accuracy curve of optimization with Baysian optimization is shown in Figure 3(b), which shows a similar results for the combination of analysis approaches. However, the performance of Baysian optimization is about 3% to 5% worse in all settings than those of RACOS .

7.3 Comparison with Other Systems

We reimplemented the sequence tagging approaches from Yang *et al.* [2009] for comparison, which uses a conditional random field to model the character and position features (denoted as CRF). We also compare our system to several online systems and evaluate the results both quantitatively and qualitatively. After excluding some systems that can not support abbreviating paper titles, we finally choose `acronymcreator.net` and `www.netsubstance.com` and denote them as sysA and sysN, respectively.

Quantitatively comparison (Recall@k)

We evaluate the previous mentioned systems on the test set (Figure 4). Our system gets the highest performance on test set. This is due to that our system employs analysis on various levels and captures information from the syntactic and semantic perspectives, which helps to find out keywords of the title so that more attention could be paid to those words. The CRF system provides worse results because it considers only the character and position information. The sysA is better only when we consider the top-1 candidate (10% of the cases). The recall of sysN is always 0 and gets omitted.

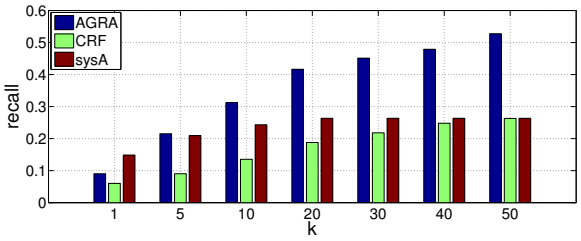


Figure 4: Recall@k of different systems.

Qualitative comparison

We use the title words of examples 2-4 of Table 2 to test the above systems and show the outputs in Table 4 for further analysis[4]. In these cases, our system finds the correct or very similar abbreviations, while the other two systems cannot.

In all cases, the results of sysN are all the random combination of letters in the title, which did not preserve the information from the title. That explains why the abbreviations given by authors never appear in sysN's results.

SysA produces a much better result than sysN. However, we notice that most of results from sysA contains the character 'a' or 'f', which is from the function word "a" and "for". The possible reason is that sysA does not remove stop words. As a result, a large portion of outputs of sysA will not be picked by human users. Also, it seems that sysA treats every input word equally without the analysis we proposed.

It is also interesting to see that in Case 3, our system successfully identifies that "wood-based composites simulations" is more important than previous words, by using the dependency tree (Figure 2). Although it misses the correct answer *WBCSim*, the candidate *EWBCSim* is very close.

Case 1	IPR: An Integrated Placement and Routing Algorithm
sysN	aagbrithui, aagoritsm, absalgvriy, alaoaithmg, aleoiithmp, aleorituma, alghrithm, alghrithma, algoelthm, algohitam, algooithm, algoritab, algorithm, algoritmyu, algorjthm, algortuml, algorwthb, algorxthm, algouitha, algouithm, algoyithop, algtrithva, algxritum, aliorishmi, aliorvthm, aluoritheq, aobting, aogoiithm, aogorioam, aogorxthm
sysA	PARA, ARA, AIR, PAR, PROA, APsARA, APART, PANDA, PARmA, PARkA, ANANDA, ARIA, PEAR, PART, PRAT, PRAM, APoLAR, ANkARA, APLANAT, PARTIAL, AlAR, AjAR, AgAR, AfAR, AdAR, APiA, AsIA, AuRA, AReA, ARAn
AGRA	IPRA, PRA, IPA, **IPR**, InPlRoA, InPRoAl, IPlRoAl, InPlRAl, InPRoA, IPlRoA, InPlRA, PlRoAl, InPlAl, InPlRo, IPRoAl, InPRAl, IPlRAl, PlRoA, IPlR, IPlA, PlRA, IPlaRoA, PlaRoAl, PlaRouA, InPRouA, InPlaRA, InPlaAl, InPlaRo, IPlRouA, PlRouAl
Case 2	SeRQL:A Second Generation RDF Query Language
sysN	absgknerat, absqueryll, absraaseco, absrdfiueu, adfrdf, ageryqnerd, aueryconnw, cdflangunm, coasecoid, comrdfseec, coyrdfrdf, eaaneuaoeu, easeocnd, edfgenkrkt, edfsecona, elanguaee, enarefgeer, entryfsecm, eqcoodseco, esejondaec, esery, exquery, exqueyy, freshrdfge, frhshgener, fteseoefse, furshsecon, gdnerhaion, geaeiatioi, gederation
sysA	ASGARD, SEQUEL, AGAR, AGAL, GIRL, SEQUELA, ARyL, ARiL, AGRa, AStER, AShER, AuGER, AnGER, AlGER, AnGEL, mARL, jARL, eARL, cARL, mSGR, AQUiLA, ASLANt, SErGER, SEeGER, SQUEaL, AGILE, GNARL, tASER, mASER, lASER
AGRA	SGRQL, SRQL, GRQL, SGQL, **SeRQL**, SGRQ, SGQ, SGR, SGL, GQL, GRL, GRQ, SQL, SRL, SRQ, RQL, SeGeQuL, SeGeRQL, SGeRQuL, SeGRQuL, SeGeRQu, SeGeRDQ, SeGeRDL, SeGeRLa, SeGeQLa, SeRQuLa, GeRQuLa, SeGQuLa, SGeQuLa, SeRDQuL, GeRDQuL
Case 3	WBCSim: A Prototype Problem Solving Environment for Wood-Based Composites Simulations
sysN	aewiroblem, aorpyotoiy, bsimdlaago, ckmenviron, comooirtes, comphsiwes, composdtfs, composgtrs, composioes, composites, compositks, comppsptea, comprsxtfh, comurobyem, comwoodbas, comyosites, comyosxtes, coupvsstes, coxkosites, eaolving, earoblxm, eeolying, eniironmen, entirqbiem, entsolving, envioonmen, environmee, environmen, envirormen, envirxnmfn
sysA	APPOSE, APPRESS, APPRiSE, PREFOCuS, APPEaSE, AEROBiCS, APPEStAT, APPRaiSE, SkEWBACk, APPOSItE, APPLauSE, PiPEFiSh, pAnPiPES, APPLE, APSIS, APPEL, APPEND, APROPOS, APPROVE, PESEWA, APPLET, APIECE, APICES, APPEaR, APPEAl, AsSESS, PREFaB, AsPECT, PaRSEC, ARSENIC
AGRA	PECoS, ProPS, ProSE, PECS, SEBS, PPS, PSE, PPE, PES, SES, PPSE, PSES, PPES, PSEW, PPEW, PPSES, PPSEW, PSEWS, PPEWS, PPSEC, PPSEWS, PSEnvi, PPEnvi, PPSECS, PPSEWC, PPSEnvi, PEBCSim, SEBCSim, **EWBCSim**, PPSEWBC

Table 4: Example outputs of different systems (Top 30). The correct results are marked with a underline; the results in bold are similar with the correct ones.

8 Related Work

Creativity is the most crucial and interesting point of the naming problem. Özbal and Strapparava [2012] used a computational approach for the task of business naming where neologism are generated based on a given category and several words as descriptions. Their method mainly works at word level, without sentence level analysis. Different from their deterministic approach, we propose a more flexible framework that is parameterized and could be optimized with data.

There are research focus on the abbreviations of daily Chinese text, which is less creative and could be handled by character level features [Chang and Lai, 2004; Yang *et al.*, 2009].

There are several other online naming systems working in different scenarios. `naming.net` requires user to specify keywords and prefixes of these words, and generates names by combining these prefixes, which does not including the analysis part. `business-name-generators.com` generates purely random business names for users, which cannot be related to a given description.

9 Conclusion

For academic papers, a good abbreviation for the title not only is easier to be referred to and be communicated, but also reminds people about the main idea or concept of the paper. This paper proposes an AI task for automatically generating abbreviations from the title word. We create data sets and propose a parameterized framework for solving the problem. By employing analysis approaches at various levels, our system obtains results that are closer to those generated by humans.

Currently, our system only deals with simple abbreviations without character replacement. We do not consider sophisticated language phenomena such as metaphor, nor do we use the information from other parts of the paper, such as the abstract or the main body. These are interesting topics to be explored as the next step.

[4]For simplicity we only show the first 30 abbreviations.

References

[Andersen *et al.*, 2008] David G. Andersen, Jason Franklin, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. FAWN: A fast array of wimpy nodes. *Communications of the ACM*, 54(7):101–109, 2008.

[Breiman, 1996] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996.

[Broekstra and Kampman, 2003] Jeen Broekstra and Arjohn Kampman. Serql: a second generation rdf query language. In *Proc. SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, pages 13–14, 2003.

[Chang and Lai, 2004] Jing Shin Chang and Yu Tso Lai. A preliminary study on probabilistic models for chinese abbreviations. In *Proceedings of the Third SIGHAN Workshop on Chinese Language Learning*, pages 9–16, 2004.

[Chen and Goodman, 2010] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 2010.

[Chen and Manning, 2014] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[Goel *et al.*, 1998] Amit Goel, Constantinos Phanouriou, Frederick A. Kamke, Calvin J. Ribbens, Clifford A. Shaffer, and Layne T. Watson. Wbcsim: A prototype problem solving environment for wood-based composites simulations. Technical report, Blacksburg, VA, USA, 1998.

[Hu *et al.*, 2017] Yi-Qi Hu, Hong Qian, and Yang Yu. Sequential classification-based optimization for direct policy search. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 2029–2035, San Francisco, CA, 2017.

[Manning *et al.*, 2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[Martinez-Cantin, 2014] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15:3915–3919, 2014.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[Miller, 1995] George Armitage Miller. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41, 1995.

[Özbal and Strapparava, 2012] Gözde Özbal and Carlo Strapparava. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 703–711, Stroudsburg, PA, 2012.

[Pan and Chu, 2007] Min Pan and Chris Chu. Ipr: An integrated placement and routing algorithm. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 59–62, New York, NY, USA, 2007. ACM.

[Stolcke, 2004] Andreas Stolcke. Srilm – An extensible language modeling toolkit. In *Proceedings of the 2004 International Conference on Spoken Language Processing*, pages 901–904, 2004.

[Wang *et al.*, 2000] James Z. Wang, Jia Li, and Gio Wiederholdy. *SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries*, pages 360–371. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[Yang *et al.*, 2009] Dong Yang, Yi-cheng Pan, and Sadaoki Furui. Automatic chinese abbreviation generation using conditional random field. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 273–276, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Yu *et al.*, 2016] Yang Yu, Hong Qian, and Yi Qi Hu. Derivative-free optimization via classification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2000–2006, Phoenix, AZ, 2016.