*entropy*

*Article*

# A Formal Model for Semantic Computing Based on Generalized Probabilistic Automata

**Guangjian Huang, Shahbaz Hassan Wasti, Lina Wei and Yuncheng Jiang** *

School of Computer Science, South China Normal University, Guangzhou 510631, China;
2017010160@m.scnu.edu.cn (G.H.); shahbazwasti@gmail.com (S.H.W.); 2018010188@m.scnu.edu.cn (L.W.)
* Correspondence: ycjiang@scnu.edu.cn

Received: 25 July 2019; Accepted: 10 September 2019; Published: 17 September 2019

**Abstract:** In most previous research, "semantic computing" refers to computational implementations of semantic reasoning. It lacks support from the formal theory of computation. To provide solid foundations for semantic computing, researchers propose a different understanding of semantic computing based on finite automata. This approach provides a computer theoretical approach to semantic computing. But finite automata are not capable enough to deal with imprecise knowledge. Therefore, in this paper, we provide foundations for semantic computing based on probabilistic automata. Even though traditional probabilistic automata can handle imprecise knowledge, their limitation resides in their being defined on a fixed finite input alphabet. This deeply restricts the abilities of automata. In this paper, we rebuild traditional probabilistic automata for semantic computing. Furthermore, our new probabilistic automata are robust enough to handle any alphabet as input. They have better performances in many applications. We provide an application for weather forecasting, a domain for which traditional probabilistic automata are not effective due to their finite input alphabet. Our new probabilistic automata can overcome these limitations.

**Keywords:** probabilistic automata; probability distribution; related concept; semantic computing

## 1. Introduction

Semantic similarity is applied in plentiful applications about artificial intelligence and computational linguistics, such as word sense disambiguation, information retrieval, knowledge acquisition and natural language processing [1–3]. But most research on Semantic Computing (SC) are devoted to the following three fields: (1) formal models for theoretical foundations of SC, such as description logics, ontology reasoning and answer set programming [4,5]; (2) fundamental languages and technologies for interoperability and reuse of information including RDF, RDFS, the OWL family of languages, the WSML family of languages, and SPARQL [6,7]; (3) some applications of SC [8–10]. As a result, "semantic computing" refers to computational implementations of semantic reasoning (e.g., ontology reasoning, rule reasoning, semantic query and semantic search) but it is not from the perspective of the formal theory of computation. To provide solid foundations for semantic computing, it is significant to bridge the gap between semantic computing and the formal theory of computation.

Recently, Reference [11] proposed a different understanding of semantic computing based on finite automata. Unlike previous research, it explains semantic computing from the computation theory perspective. Their models for semantic computing are based on finite automata. However, finite automata cannot deal with imprecise knowledge, which is the advantage of probabilistic automata. Unfortunately, traditional probabilistic automaton $M = (Q, \Sigma, \delta, q_0, F)$ is not capable enough, because it is defined on a finite input alphabet $\Sigma$. When an undefined input $a \notin \Sigma$ is transmitted to automaton in the applications, it will be invalid. But in the applications, the inputs transmitted from users are unpredictable. Users may input synonyms, equivalent concepts, acronyms or even some words with spelling errors. The fixed finite set of inputs restrict the abilities of automata deeply. For robustness, we prefer to build an automaton which can take any alphabet as input and semantic computing is the key to build such automata.

In this paper, we rebuild traditional probabilistic automata to provide formal models for semantic computing, based on semantic similarity of two symbols or concepts [2,3,12–14]. We will apply the existing methods in semantic similarity to probabilistic automata directly. For more information about semantic similarity, readers are encouraged to read our previous work [2,3,15].

The intuitive idea to rebuild a traditional probabilistic automaton is that we redefine the string $S \notin \Sigma$ as its corresponding most similar input string $S' \in \Sigma$, by the methods of semantic similarity. Specifically, when automata need to deal with an undefined input string $S$, we find its similar string $S'$ in the input alphabet. Then we handle string $S$ in the way similar to $S'$. As a result, probabilistic automata can take any string as input, whether defined or not. In the following section, we will propose an application of the weather forecast. Because the weather factor has been changing all the time, it is impossible to define all the weather factors in a finite input alphabet. Therefore, traditional probabilistic automata are not competent in the weather forecast. But with the methods in semantic computing, for any undefined input, our new models can take them as corresponding most similar defined inputs. Moreover, our new models provide a computer theoretical approach to semantic computing.

The rest of this paper is organized as follows. In Section 2, we briefly recall some basic notions of probabilistic automata. In Section 3, we build the models for semantic computing based on probabilistic automata by semantic similarity. Section 4 gives an application for the weather forecast, and Section 5 concludes this paper.

## 2. Related Work

For convenience, in this section, we will briefly recall some basic notions of probabilistic automata. There are more details in References [16–20].

**Definition 1** (Probability distribution)**.** *Suppose $U$ is a finite set. A probability distribution on $U$ is a function $f$ from $U$ to [0,1], if $\sum_{x\in U} f(x) = 1$. The set $\{x \in U | f(x) > 0\}$ is called the support of $U$. The set of all probability distributions on $U$ is denoted as $D(U)$. A probability distribution $f \in D(U)$ is denoted as follows [21]:*

$$f = f(x_1)/x_1 + f(x_2)/x_2 + f(x_3)/x_3 \cdots .$$

*For any $\lambda \in [0,1], f \in D(U)$, scalar multiplication $\lambda \cdot f : U \to [0,1]$ is defined as $(\lambda \cdot f)(x) = \lambda \cdot f(x)$.*

**Definition 2** (PA)**.** *A probabilistic automaton (PA) is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where*

- *$Q$ is a finite set of states,*
- *$\Sigma$ is a finite input alphabet,*
- *$\delta : Q \times \Sigma \to D(Q)$ is the transition function,*
- *$q_0 \in Q$ is the start state, $q_0 = \sum_i \alpha_i q_i$ with $\sum \alpha_i = 1, a_i \in [0,1]$,*
- *$F \subseteq Q$ is the set of accept states.*

**Remark 1.** *$\delta(q,a)(p)$ denotes the probability of automaton entering state $p$ with input $a$ from state $q$. In [17], transition function is defined in an equivalent form $\delta' : Q \times \Sigma \times Q \to [0,1]$ satisfying $\sum_{p\in Q} \delta'(q,a,p) = 1$. $\delta'(q,a,p)$ presents the probability of automaton entering state $p$ with input $a$ from state $q$, i.e., $\delta(q,a)(p) = \delta'(q,a,p)$. In the form $\delta : Q \times \Sigma \to D(Q)$, we do not have a constraint similar to $\sum_{p\in Q} \delta'(q,a,p) = 1$, because by complete probability formula we can get that for any $f \in D(U), \sum f(x) = 1$. For convenience, both forms of transition function are denoted as $\delta$, i.e., $\delta(q,a)(p) = \delta(q,a,p)$.*

**Definition 3** (Semantic similarity)**.** *Let $U$ be a set of concepts or symbols. The semantic similarity of two concepts $a, b \in U$ is defined as $sim : U \times U \to I$, where $I$ is unit interval.*

In traditional approaches, Reference [1] computes semantic similarity of concepts based on WordNet. Our previous work [2,3,15] is based on Wikipedia. Couto, F.M. et al. [22] proposed an approach by ontology. In this paper, we combine the theory of semantic similarity and computation to construct a new model of probabilistic automaton.

## 3. Formal Model of Semantic Computing Based on Probabilistic Automata

In this section, we present formal models for semantic computing by generalizing the automata mentioned in the above section. Firstly, we consider a simple case, extending automata by equivalent concepts. Then we investigate a general situation, generalizing automata by semantically related concepts.

### *3.1. Probabilistic Automata under Equivalent Concepts*

Suppose that $M = (Q, \Sigma, \delta, q_0, F)$ is a PA. One of the limitation of a PA $M$ is that $M$ is restricted by finite input alphabet $\Sigma$. Suppose $x = a_1a_2 \cdots a_n \in \Sigma^*$ ($\Sigma^*$ is the set of all strings over $\Sigma$, containing empty string $\varepsilon$) and $x' = a_1'a_2' \cdots a_n' \notin \Sigma^*$. For any $1 \le i \le n$, $a_i$ and $a_i'$ are equivalent concept or synonym in semantics. For instance, $x = a_1a_2 \in \Sigma^*$ and $x' = a_1'a_2' \notin \Sigma^*$ where $a_1 = true, a_2 = valid$ and $a_1' = correct, a_2' = authorized$. $M$ is valid with input $x$, but not $x'$. But the question is that in many applications, the inputs transmitted from users are unpredictable. Previously, we defined $a_1 = true$ as a legal input. But in the application, users may input $a_1' = correct$ instead of $a_1 = true$. Because $a_1' \notin \Sigma$, automaton $M$ is invalid with input $a_1'$. For robustness, the automaton $M$ is supposed to have the ability to take equivalent concepts or synonyms as inputs as well.

**Definition 4.** *Suppose $\Sigma$ is an alphabet and $\pi$ is the alphabet of all the possible symbols. For any $b \in \Sigma, a \in \pi$, if $sim(a, b) = 1$, then $b$ is called an equivalent concept or synonym of $a$ in $\Sigma$, denoted as $a^e = b$. If $a \in \Sigma$ then $a^e = a$. If there does not exist any equivalent concept or synonym in $\Sigma$, then $a^e = \varepsilon$ where $\varepsilon$ is empty string. Therefore, for any $a \in \pi$, there exist $a^e \in \Sigma^*$. Both equivalent concepts and synonyms are took as equivalent concepts in this paper.*

In the case of $a^e = \varepsilon$, the transition function $\delta : Q \times \Sigma \to D(Q)$ of a PA $M$ need to be generalized to a function $Q \times \Sigma^* \to D(Q)$ and for convenience, we still denote it as $\delta$. For $q \in Q$,

$$\delta(q, A) = \begin{cases} \delta(q, \varepsilon) = 1/q, & if \;\; A = \varepsilon, \\ \delta(q, a_1a_2) = \sum_{p \in Q} \delta(q, a_1)(p) \cdot \delta(p, a_2), & if \;\; A = a_1a_2 \;\text{ and }\; a_1 \in \Sigma^*, a_2 \in \Sigma. \end{cases}$$

The language accepted by PA $M$ is defined as a function $L_M : \Sigma^* \to [0, 1]$: for any $x = a_1a_2 \cdots a_n \in \Sigma^*$,

$$L_M(x) = \sum_{q \in F, q_i \in Q} \{\delta(q_0, a_1, q_1) \cdot \delta(q_1, a_2, q_2) \cdots \delta(q_{n-1}, a_n, q)\}, i = 1, 2 \cdots n-1,$$

where $\delta(q_0, a_1, q_1) = \sum_i \alpha_i \delta(q_i, a_1, q_1)$, if $q_0 = \sum_i \alpha_i q_i$.

**Definition 5** (DPEC)**.** *A deterministic probabilistic automaton for semantic computing under equivalent concepts (DPEC) is a seven-tuple $M = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, where*

- *$Q$ is a finite set of states,*
- *$\Sigma$ is a finite input alphabet,*
- *$\delta : Q \times \Sigma \to D(Q)$ is a internal transition function,*
- *$q_0 \in Q$ is the start state, $q_0 = \sum_i \alpha_i q_i$ with $\sum \alpha_i = 1, a_i \in [0, 1]$,*
- *$F \subseteq Q$ is the set of accept states,*
- *$\Sigma' = \{a \mid \exists! a^e \in \Sigma, a \in \pi\}$, where $\exists! a^e \in \Sigma$ means there exist only one $a^e \in \Sigma$ for every $a$,*

- $\delta' : Q \times (\Sigma \cup \Sigma') \to D(Q)$ *is a generalized transition function of* $\delta$*:*

$$\delta'(q,a) = \begin{cases} \delta(q,a), & if \ \ a \in \Sigma, \\ \delta(q,a^e), & if \ \ a \in \Sigma'. \end{cases}$$

In the traditional perspective, $\Sigma \cup \Sigma'$ is the alphabet of $M$ and $\delta'$ is the transition function of $M$. But in this paper, the transition function $\delta$ is called internal transition function and $\delta'$ is called generalized transition function. Simply, by transition function of $M$, we mean $\delta'$. An input in $\Sigma$ is called an original input and an input in $\Sigma'$ is called a generalized input.

**Example 1** (Command order). *For instance, we consider a sample case of command order. We consider a simple case that every time users input a command, automaton will enter a state based on probability distribution. Suppose that in PA* $M = (Q, \Sigma, \delta, q_0, F)$*,* $Q = \{q, p\}$*.* $\Sigma = \{true, false\}$*.* $q_0 = q, F = Q$*.* $\delta$ *is defined as:*

$$\begin{aligned} &\delta(q, true, q) = 0.7, \delta(q, false, q) = 0.8, \\ &\delta(q, true, p) = 0.3, \delta(q, false, p) = 0.2, \\ &\delta(p, true, q) = 0.1, \delta(p, false, q) = 0.4, \\ &\delta(p, true, p) = 0.9, \delta(p, false, p) = 0.6. \end{aligned}$$

*In PA M, "true" and "false" in* $\Sigma$ *are commands input by users.* $\delta(q, true, p) = 0.3$ *means when users are in state q and input command "true", the probability of that he will enter state p is 0.3. Others are similar. But in application, users may input "right" and "wrong" instead of "true" and "false", because they are equivalent concepts for users. In our daily life, there are many equivalent concepts and users cannot know which are legal commands.*

*In order to empower M with the ability of taking equivalent concepts as commands, we rebuild PA M as a DPEC* $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ *as follows. Suppose that we need to consider the additional case that users will input* "wrong" *and* "right"*, i.e.,* $\Sigma' = \{wrong, right\}$*. For other cases, it is analogous. Define* $\delta'(k, a, k') = \delta(k, a, k')$*, for* $k, k' \in Q, a \in \Sigma$*. Suppose*

$$sim(right, true) = 1, sim(wrong, false) = 1,$$

*and for other cases* $sim(a, a') = 0, a' \in \Sigma', a \in \Sigma$*. For* $a \in \Sigma'$*,* $\delta'$ *is defined as:*

$$\begin{aligned} &\delta'(q, right, p) = \delta(q, true, p) = 0.3, \\ &\delta'(q, wrong, p) = \delta(q, false, p) = 0.2, \\ &\delta'(q, right, q) = \delta(q, true, q) = 0.7, \\ &\delta'(q, wrong, q) = \delta(q, false, q) = 0.8, \\ &\delta'(p, right, q) = \delta(p, true, q) = 0.1, \\ &\delta'(p, wrong, q) = \delta(p, false, q) = 0.4, \\ &\delta'(p, right, p) = \delta(p, true, p) = 0.9, \\ &\delta'(p, wrong, p) = \delta(p, false, p) = 0.6. \end{aligned}$$

*With the above extended* $\Sigma'$ *and* $\delta'$*, when we suppose to get "true" from users, but users input "right", PA M is invalid. However, DPEC* $M'$ *is still valid. Similarly, we can deal with spelling error, such as "ture". The key to do this is the semantic similarity function sim. If we define* $sim(true, ture) = 1$ *and* $\delta'(k, ture, k') = \delta'(k, true, k'), k, k' \in Q$*, then spelling error "ture" can be taken as a legal input.*

But in many cases, for some $a \in \pi$, there may exist several $a^e \in \Sigma$. This is defined as the following nondeterministic case.

**Definition 6** (NPEC). *A nondeterministic probabilistic automaton for semantic computing under equivalent concepts (NPEC) is a seven-tuple* $M = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$*, where*

- *$Q, \Sigma, \delta, q_0, F$ are the same in DPEC,*
- *$\Sigma' = \{a | a \in \pi, \exists a^e \in \Sigma\}$,*
- *$\delta' : Q \times (\Sigma \cup \Sigma') \to D(Q)$ is a generalized transition function of $\delta$:*

$$\delta'(q,a) = \begin{cases} \delta(q,a), \ if \ a \in \Sigma, \\ \sum_{a^e \in \Sigma} \delta(q,a^e), \ if \ \sum_{a^e \in \Sigma} \delta(q,a^e) \leq 1, a \in \Sigma', \\ 1, \ if \ \sum_{a^e \in \Sigma} \delta(q,a^e) > 1, a \in \Sigma'. \end{cases}$$

In the traditional perspective, $\Sigma \cup \Sigma'$ is the alphabet of NPEC $M$ and $\delta'$ is the transition function of NPEC $M$.

**Example 2** (Command order). *Suppose that the designers of the PA M in above Example 1 get the feedback that users get an error report with inputs "right" and "wrong" frequently. In order to make it more convenient for users, the designers improve it as follows. In PA $M = (Q, \Sigma, \delta, q_0, F)$, $\Sigma$ is redefined as $\Sigma = \{true, false, right, wrong\}$. $\delta$ is redefined as:*

$$\begin{aligned} &\delta(q, right, p) = \delta(q, true, p) = 0.3, \\ &\delta(q, wrong, p) = \delta(q, false, p) = 0.2, \\ &\delta(q, right, q) = \delta(q, true, q) = 0.7, \\ &\delta(q, wrong, q) = \delta(q, false, q) = 0.8, \\ &\delta(p, right, q) = \delta(p, true, q) = 0.1, \\ &\delta(p, wrong, q) = \delta(p, false, q) = 0.4, \\ &\delta(p, right, p) = \delta(p, true, p) = 0.9, \\ &\delta(p, wrong, p) = \delta(p, false, p) = 0.6. \end{aligned}$$

*But there are so many equivalent concepts. What if some users may input "correct" instead of "right" or "true"?*

*In this case, we rebuild PA M as a NPEC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ as follows. In this case, $\Sigma' = \{correct\}$. Define $\delta'(k, a, k') = \delta(k, a, k')$, for $k, k' \in Q, a \in \Sigma$. Because $\delta(q, right, p) = \delta(q, true, p) = 0.3$ and "correct" is equivalent to "right" and "true", so we define $\delta'(q, correct, p) = \delta(q, right, p) + \delta(q, true, p) = 0.3 + 0.3 = 0.6$. Analogously, $\delta'$ can be defined as:*

$$\begin{aligned} &\delta'(q, right, p) = \delta(q, right, p) + \delta(q, true, p) = 0.6, \\ &\delta'(q, right, q) = min\{\delta(q, right, q) + \delta(q, true, q), 1\} = 1, \\ &\delta'(p, right, q) = \delta(p, right, q) + \delta(p, true, q) = 0.2, \\ &\delta'(p, right, p) = min\{\delta(p, right, q) + \delta(p, true, p), 1\} = 1. \end{aligned}$$

*With the above extended $\Sigma'$ and $\delta'$, when we suppose to get "true" and "right" from users, but users input "correct", PA M is invalid. However, NPEC $M'$ is still valid. Similarly, we can deal with other equivalent concepts.*

The key point is that the inputs transmitted from users are unpredictable. The finite input alphabet $\Sigma$ in PA $M$ cannot take every case into account. But NPEC is competent. Based on the research on semantic computing introduced previously, we can apply their methods of semantic computing to PA $M$ directly. Once an undefined input is transmitted to $M$, we find its equivalent inputs and modify $\delta'$, then it can be taken as a legal input. But most of the time, we can only find similar inputs instead of equivalent concepts. We leave this case to the next section.

The transition function of a DPEC or NPEC $M'$ can be generalized to a function $\delta' : Q \times (\Sigma \cup \Sigma')^* \to D(Q)$ and for convenience, we still denote it as $\delta'$:

$$\delta'(q, A) = \begin{cases} \delta'(q, \varepsilon) = 1/q, \ \ if \ \ A = \varepsilon, q \in Q, \\ \delta'(q, a_1a_2) = \sum_{p \in Q} \delta'(q, a_1)(p) \cdot \delta'(p, a_2), \ \ if \ \ A = a_1a_2 \ \ and \ \ a_1 \in (\Sigma \cup \Sigma')^*, \\ a_2 \in \Sigma \cup \Sigma', q \in Q. \end{cases}$$

By the above generalized transition function, we can get the robustness of a DPEC or NPEC $M'$:

**Theorem 1** (Robustness)**.** *Suppose that $M'$ is a DPEC or NPEC, then $M'$ can take any string of symbols $A \in \pi^*$ as an input, i.e., the transition function of a DPEC or NPEC $M'$ can be generalized to a function $\delta' : Q \times \pi^* \to D(Q)$.*

**Proof.** For any $q \in Q$, if $A = \varepsilon$, then $\delta'(q, A) = \delta'(q, \varepsilon) = 1/q$. If $A \neq \varepsilon$ then suppose $A = a_1a_2 \cdots a_n$ where $a_i \in \pi, 1 \leq i \leq n$. If $a_i = \varepsilon$, then $\delta'(q, a_i) = \delta'(q, \varepsilon) = 1/q$. If $a_i \in \Sigma$, then $\delta'(q, a_i) = \delta(q, a_i)$. If $a_i \notin \Sigma$, then $\delta'(q, a_i) = \delta(q, a_i^e)$. Recall that if $a \in \Sigma$, then $a^e = a$. Therefore

$$\begin{aligned} &\delta'(q, A) \\ &= \delta'(q, a_1a_2 \cdots a_n) \\ &= \textstyle\sum_{p_1p_2 \cdots p_{n-1} \in Q} \delta'(q, a_1)(p_1) \cdot \delta'(p_1, a_2)(p_2) \cdots \delta'(p_{n-1}, a_n) \\ &= \textstyle\sum_{p_1p_2 \cdots p_{n-1} \in Q} \delta(q, a_1^e)(p_1) \cdot \delta(p_1, a_2^e)(p_2) \cdots \delta(p_{n-1}, a_n^e), \end{aligned}$$

which means $M'$ can take any string of symbols $A \in \pi^*$ as an input, i.e., the transition function of $M'$ can be generalized to a function $\delta' : Q \times \pi^* \to D(Q)$. □

As analysed above, in many applications of automata, the inputs transmitted from users are unpredictable. If an automaton can take any string of symbols as an input, it can be applied to more fields. The robustness of probabilistic automata empowers it to be more stable. Intuitively, a DPEC or NPEC is a semantic expansion of PA which releases PA from a finite input alphabet.

**Definition 7.** *The language accepted by a DPEC or NPEC $M'$ is defined as a function $L_{M'} : \pi^* \to [0, 1]$: for any $x = a_1a_2 \cdots a_n \in \pi^*$,*

$$L_{M'}(x) = \sum_{q \in F, q_i \in Q} \{\delta'(q_0, a_1, q_1) \cdot \delta'(q_1, a_2, q_2) \cdots \delta'(q_{n-1}, a_n, q)\}, i = 1, 2 \cdots n-1.$$

DPEC and NPEC are semantic generalizations of PA for semantic computing. To show this relationship, we can define DPEC and NPEC in the following equivalent forms.

**Definition 8** (DPEC)**.** *A deterministic probabilistic automaton for semantic computing under equivalent concepts (DPEC) is a seven-tuple $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, where*

- *$M = (Q, \Sigma, \delta, q_0, F)$ is a PA,*
- *$\Sigma' = \{a | a \in \pi, \exists! a^e \in \Sigma\}$,*
- *$\delta' : Q \times (\Sigma \cup \Sigma')^* \to D(Q)$ is a generalized transition function of $\delta$:*

$$\delta'(q, a) = \begin{cases} \delta(q, a), \ \ if \ \ a \in \Sigma, \\ \delta(q, a^e), \ \ if \ \ a \in \Sigma'. \end{cases}$$

**Definition 9** (NPEC)**.** *A nondeterministic probabilistic automaton for semantic computing under equivalent concepts (NPEC) is a seven-tuple $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, where*

- *$M = (Q, \Sigma, \delta, q_0, F)$ is a PA,*
- *$\Sigma' = \{a | \exists a^e \in \Sigma, a \in \pi\}$,*

- $\delta' : Q \times (\Sigma \cup \Sigma')^* \to D(Q)$ *is a generalized transition function of* $\delta$:

$$\delta'(q,a) = \begin{cases} \delta(q,a), \ if \ \ a \in \Sigma, \\ \sum_{a^e \in \Sigma} \delta(q,a^e), \ if \ \ \sum_{a^e \in \Sigma} \delta(q,a^e) \le 1, a \in \Sigma', \\ 1, \ if \ \ \sum_{a^e \in \Sigma} \delta(q,a^e) > 1, a \in \Sigma'. \end{cases}$$

The language accepted by DPEC (or NPEC) and PA have the following property.

**Theorem 2** (Semantic generalization). *Suppose that DPEC* $M' = (Q,\Sigma,\delta,\Sigma',\delta',q_0,F)$ *is a semantic generalizationof PA* $M = (Q,\Sigma,\delta,q_0,F)$ *and the language accepted by them are function* $L_{M'}$ *and* $L_M$ *respectively. The languages accepted by them have the following properties:*

1. *for any* $x \in \Sigma^*$, $L_{M'}(x) = L_M(x)$,
2. *for any* $x \in \pi^*$, *there exist* $x' \in \Sigma^*$ *such that* $L_{M'}(x) = L_{M'}(x') = L_M(x')$.

**Proof.** For DPEC $M'$ and PA $M$, if $x = \varepsilon$, then $\delta'(q,x) = \delta(q,x) = 1/q$, for any $q \in Q$. Obviously, $L_{M'}(x) = L_M(x)$. If $x \neq \varepsilon$, (1) for any $x = a_1a_2\cdots a_n \in \Sigma^*, a_i \in \Sigma, 1 \le i \le n-1$,

$$L_M(x) = \sum_{q \in F, q_i \in Q} \{\delta(q_0,a_1,q_1)\cdot\delta(q_1,a_2,q_2)\cdots\delta(q_{n-1},a_n,q)\}.$$

Since $\delta(q,a,p) = \delta'(q,a,p)$, for any $a \in \Sigma, q, p \in Q$, we get

$$\begin{aligned} & L_{M'}(x) \\ & = \textstyle\sum_{q \in F, q_i \in Q}\{\delta'(q_0,a_1,q_1)\cdot\delta'(q_1,a_2,q_2)\cdots\delta'(q_{n-1},a_n,q)\} \\ & = \textstyle\sum_{q \in F, q_i \in Q}\{\delta(q_0,a_1,q_1)\cdot\delta(q_1,a_2,q_2)\cdots\delta(q_{n-1},a_n,q)\} \\ & = L_M(x). \end{aligned}$$

(2) For any $x = a_1a_2\cdots a_n \in \pi^*, a_i \in \pi, 1 \le i \le n-1$,

$$L_{M'}(x) = \sum_{q \in F, q_i \in Q} \{\delta'(q_0,a_1,q_1)\cdot\delta'(q_1,a_2,q_2)\cdots\delta'(q_{n-1},a_n,q)\}.$$

For $a \in \Sigma'$, $\delta'(q,a) = \delta(q,a^e)$ and recall that $a^e = a$ if $a \in \Sigma$. Let $x' = a_1^e a_2^e \cdots a_n^e$, then $x' \in \Sigma^*$.

$$\begin{aligned} & L_{M'}(x) \\ & = \textstyle\sum_{q \in F, q_i \in Q}\{\delta'(q_0,a_1,q_1)\cdot\delta'(q_1,a_2,q_2)\cdots\delta'(q_{n-1},a_n,q)\} \\ & = \textstyle\sum_{q \in F, q_i \in Q}\{\delta(q_0,a_1^e,q_1)\cdot\delta(q_1,a_2^e,q_2)\cdots\delta(q_{n-1},a_n^e,q)\} \\ & = L_{M'}(x'). \end{aligned}$$

Since $x' \in \Sigma^*$, by (1) we can get $L_{M'}(x') = L_M(x')$. Therefore, $L_{M'}(x) = L_{M'}(x') = L_M(x')$. □

**Theorem 3** (Semantic generalization). *Suppose that NPEC* $M' = (Q,\Sigma,\delta,\Sigma',\delta',q_0,F)$ *is a semantic generalization of PA* $M = (Q,\Sigma,\delta,q_0,F)$ *and the language accepted by them are function* $L_{M'}$ *and* $L_M$ *respectively. The languages accepted by them have the following properties:*

1. *for any* $x \in \Sigma^*$, $L_{M'}(x) = L_M(x)$,
2. *for any* $x \in \pi^*$, *there exist* $x' \in \Sigma^*$ *such that* $L_{M'}(x) \ge L_{M'}(x') = L_M(x')$.

**Proof.** For NPEC $M'$ and PA $M$, if $x = \varepsilon$ or $x = a_1a_2\cdots a_n \in \Sigma^*, a_i \in \Sigma, 1 \le i \le n$, it is similar to DPEC.

For any $x = a_1 a_2 \cdots a_n \in \pi, a_i \in \pi, 1 \leq i \leq n-1$,

$$L_{M'}(x) = \sum_{q \in F, q_i \in Q} \{\delta'(q_0, a_1, q_1) \cdot \delta'(q_1, a_2, q_2) \cdots \delta'(q_{n-1}, a_n, q)\}, i = 1, 2 \cdots n-1.$$

For $a \in \Sigma'$, $\delta(q, a^e, p) \leq \sum_{a^e \in \Sigma} \delta(q, a^e, p)$ and $\delta(q, a^e, p) \leq 1$, therefore,

$$\delta'(q, a, p) = min\{\sum_{a^e \in \Sigma} \delta(q, a^e, p), 1\} \geq \delta(q, a^e, p).$$

Recall that $a^e = a$ if $a \in \Sigma$. Let $x' = a_1^e a_2^e \cdots a_n^e$, then $x' \in \Sigma^*$.

$$\begin{aligned} & L_{M'}(x) \\ & = \textstyle\sum_{q \in F, q_i \in Q} \{\delta'(q_0, a_1, q_1) \cdot \delta'(q_1, a_2, q_2) \cdots \delta'(q_{n-1}, a_n, q)\} \\ & \geq \textstyle\sum_{q \in F, q_i \in Q} \{\delta(q_0, a_1^e, q_1) \cdot \delta(q_1, a_2^e, q_2) \cdots \delta(q_{n-1}, a_n^e, q)\} \\ & = L_{M'}(x'). \end{aligned}$$

Since $x' \in \Sigma^*$, by (1) we can get $L_{M'}(x') = L_M(x')$. Therefore, $L_{M'}(x) \geq L_{M'}(x') = L_M(x')$. □

The intuitive idea of the above two theorems is that because DPEC and NPEC $M'$ are semantically generalized from PA $M$, then for an original input in $\Sigma$, they will get the same result. For a generalized input in $\Sigma'$, if there is only one equivalent concept, they will get an equivalent result too. But if there are several equivalent concepts, the generalized input will include all the equivalent cases, then NPEC $M'$ will get a bigger probability. The properties of semantic generalization can be reflected by their transition functions and the languages accepted.

An important difference between DPEC and NPEC is that a DPEC is a traditional probabilistic automaton but an NPEC $M'$ is not a traditional probabilistic automaton because it does not satisfy complete probability formula. In PA $M$, $\sum_{p \in Q} \delta(q, a, p) = 1$. But in corresponding NPEC $M'$, because $\delta'(q, a, p) \geq 0$, when $a \in \Sigma'$,

$$\sum_{p \in Q} \delta'(q, a, p) \geq \sum_{p \in Q} \delta(q, a, p) = 1.$$

If we want to define a NPEC $M'$ as a kind of traditional probabilistic automaton, the transition function needs to be adjusted as follows.

**Definition 10** (NPEC). *A nondeterministic probabilistic automaton for semantic computing under equivalent concepts (NPEC) is a seven-tuple $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, where*

- *$M = (Q, \Sigma, \delta, q_0, F)$ is a PA,*
- *$\Sigma' = \{a | a \in \pi, \exists a^e \in \Sigma\}$, $N(a^e)$ is the number of the equivalent concepts of a,*
- *$\delta' : Q \times (\Sigma \cup \Sigma')^* \to D(Q)$ is a generalized transition function of $\delta$:*

$$\delta'(q, a) = \begin{cases} \delta(q, a), & if \ \ a \in \Sigma, \\ \frac{1}{N(a^e)} \sum_{a^e \in \Sigma} \delta(q, a^e), & if \ \ a \in \Sigma'. \end{cases}$$

It is obvious that the properties of robustness and semantic generalization are still valid and by the above adjusted transition function, it is easy to get the following property.

**Theorem 4** (Semantic computing). *In a DPEC or NPEC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, for any $a \in \Sigma'$, $\sum_{p \in Q} \delta'(q, a)(p) = sim(a, a^e) = 1$.*

**Proof.** In a DPEC or NPEC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$,

1. for any $a \in \Sigma$, $\sum_{p \in Q} \delta'(q,a)(p) = \sum_{p \in Q} \delta(q,a)(p) = 1 = sim(a, a^e)$,
2. for any $a \in \Sigma'$,

$$\begin{aligned}
&\textstyle\sum_{p \in Q} \delta'(q,a)(p) \\
&= \textstyle\sum_{p \in Q} \frac{1}{N(a^e)} \sum_{a^e \in \Sigma} \delta(q, a^e)(p) \\
&= \textstyle\frac{1}{N(a^e)} \sum_{a^e \in \Sigma} \sum_{p \in Q} \delta(q, a^e)(p) \\
&= \textstyle\frac{1}{N(a^e)} \sum_{a^e \in \Sigma} 1 \\
&= \textstyle\frac{1}{N(a^e)} * N(a^e) \\
&= 1 \\
&= sim(a, a^e).
\end{aligned}$$

□

The intuitive idea of this theorem is that the total probability of that a DPEC or NPEC $M'$ will enter all the states with a similar input is the semantic similarity between generalized input and similar input. The main purpose of a DPEC (or NPEC) $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ generalized from PA $M = (Q, \Sigma, \delta, q_0, F)$ is semantic computing.

From the above theorems, we can get that all the generalized PA do not have the limitations coming from the fixed finite input alphabet. They explain semantic computing with computation theory, instead of implementations of semantic reasoning (e.g., ontology reasoning, rule reasoning, semantic query, and semantic search).

### *3.2. Probabilistic Automata under Related Concept*

In the previous subsection, we suppose that for any input $a \notin \Sigma$, there exists an $a^e \in \Sigma$. In fact, it is just an ideal case. As illustrated in bookshop example, when a PA $M = (Q, \Sigma, \delta, q_0, F)$ is applied to an application, in order to handle unpredictable inputs, we rebuild it as a NPEC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$. When an input $a \notin \Sigma$ transmitted from users, we check the original input alphabet $\Sigma$. If there exist a $a^e \in \Sigma$ such that $sim(a, a^e) = 1$, then we add $a$ to $\Sigma'$ and define $\delta'(q,a) = \delta(q, a^e), q \in Q$. But it is too hard to find a $a^e \neq \varepsilon \in \Sigma$ such that $sim(a, a^e) = 1$ for every input $a$. In fact, most of the time, we can only find a $a^* \in \Sigma$ such that $0 \leq sim(a, a^*) \leq 1$. Therefore, in this subsection, we generalize DPEC and NPEC to the case $0 \leq sim(a, a^*) \leq 1$.

**Definition 11.** *Suppose $\Sigma$ is an alphabet and $\pi$ is the alphabet of all the possible symbols. For any $b \in \Sigma, a \in \pi$, if $sim(a,b) = max_{b' \in \Sigma}\{sim(a,b')\} > 0$, then $b$ is called the most similar concepts of $a$ in $\Sigma$, denoted as $a^* = b$. If $a \in \Sigma$ then $a^* = a^e = a$. If $max_{b' \in \Sigma}\{sim(a,b')\} = 0$, then $a^* = \varepsilon$. Notice that $a^*$ is not unique.*

With the most similar concepts, we generalize DPEC and NPEC to the universal case as follows.

**Definition 12** (DPRC). *A deterministic probabilistic automaton for semantic computing under related concept (DPRC) is a seven-tuple $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, where*

- *$M = (Q, \Sigma, \delta, q_0, F)$ is a PA,*
- *$\Sigma' = \{a | a \in \pi, \exists! a^* \in \Sigma\}$,*
- *$\delta' : Q \times (\Sigma \cup \Sigma') \to D(Q)$ is a generalized transition function of $\delta$:*

$$\delta'(q,a) = \begin{cases} \delta(q,a), & if \;\; a \in \Sigma, \\ \delta(q,a^*) \cdot sim(a,a^*), & if \;\; a \in \Sigma'. \end{cases}$$

**Definition 13** (NPRC). *A nondeterministic probabilistic automaton for semantic computing under related concept (NPRC) is a seven-tuple $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$, where*

- *$M = (Q, \Sigma, \delta, q_0, F)$ is a PA,*
- *$\Sigma' = \{a | a \in \pi, \exists a^* \in \Sigma\}$,*

- $\delta' : Q \times (\Sigma \cup \Sigma') \to D(Q)$ *is a generalized transition function of* $\delta$*:*

$$\delta'(q,a) = \begin{cases} \delta(q,a), \ if \ \ a \in \Sigma, \\ \sum_{a^* \in \Sigma} \delta(q,a^*), \ if \ \ \sum_{a^* \in \Sigma} \delta(q,a^*) \leq 1, a \in \Sigma', \\ 1, \ if \ \ \sum_{a^* \in \Sigma} \delta(q,a^*) > 1, a \in \Sigma'. \end{cases}$$

Obviously, DPEC and NPEC are special cases of DPRC and NPRC, that is, $sim(a,a^*) = sim(a,a^e) = 1$.

**Example 3** (Command order)**.** *Recall the PA M in above Example 2. Here we consider the case that users may input "fit" instead of "right" or "true". The term "fit" is not equivalent concept of "right" or "true", but they are similar. Suppose* $sim(fit,right) = 0.7, sim(fit,true) = 0.8$. *Then we can only find a similar concept for "fit" instead of an equivalent concept. So DPEC and NPEC are not valid in this case. In fact, the probability of getting a similar input is bigger than an equivalent concept.*

*In order to deal with similar inputs, we rebuild PA M as a DPRC* $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ *as follows. In this case,* $\Sigma' = \{fit\}$. *Define* $\delta'(k,a,k') = \delta(k,a,k')$, *for* $k,k' \in Q, a \in \Sigma$. $\delta'$ *is defined as:*

$$\begin{aligned} \delta'(q,fit,p) &= sim(fit,true) \cdot \delta(q,true,p) = 0.24, \\ \delta'(q,fit,q) &= sim(fit,true) \cdot \delta(q,true,q) = 0.56, \\ \delta'(p,fit,q) &= sim(fit,true) \cdot \delta(p,true,q) = 0.08, \\ \delta'(p,fit,p) &= sim(fit,true) \cdot \delta(p,true,p) = 0.72. \end{aligned}$$

*With the above extended* $\Sigma'$ *and* $\delta'$*, when we suppose to get "true" and "right" from users, but users input "fit", PA M is invalid. However, DPRC* $M'$ *is still valid.*

*Suppose* $sim(fit,right) = sim(fit,true) = 0.8$*, then PA M need to be rebuild as a NPRC* $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ *as follows.* $\Sigma' = \{fit\}$. *Define* $\delta'(k,a,k') = \delta(k,a,k')$, *for* $k,k' \in Q, a \in \Sigma$. $\delta'$ *is defined as:*

$$\begin{aligned} &\delta'(q,fit,p) = sim(fit,right) \cdot \delta(q,right,p) + sim(fit,true) \cdot \delta(q,true,p) = 0.48, \\ &\delta'(q,fit,q) = min\{sim(fit,right) \cdot \delta(q,right,q) + sim(fit,true) \cdot \delta(q,true,q), 1\} = 1, \\ &\delta'(p,fit,q) = sim(fit,right) \cdot \delta(p,right,q) + sim(fit,true) \cdot \delta(p,true,q) = 0.16, \\ &\delta'(p,fit,p) = min\{sim(fit,right) \cdot \delta(p,right,q) + sim(fit,true) \cdot \delta(p,true,p), 1\} = 1. \end{aligned}$$

*Spelling errors can also be took as similar inputs analogously.*

The transition function of a DPRC or NPRC $M'$ can also be generalized to a function $\delta' : Q \times (\Sigma \cup \Sigma')^* \to D(Q)$ and for convenience, we still denote it as $\delta'$:

$$\delta'(q,A) = \begin{cases} \delta'(q,\varepsilon) = 1/q, \ if \ \ A = \varepsilon, q \in Q, \\ \delta'(q,a_1a_2) = \sum_{p \in Q} \delta'(q,a_1)(p) \cdot \delta'(p,a_2), \ if \ \ A = a_1a_2 \ \ and \ \ a_1 \in (\Sigma \cup \Sigma')^*, \\ a_2 \in \Sigma \cup \Sigma', q \in Q. \end{cases}$$

By this generalized transition function, a DPRC (or NPRC) $M$ inherits the robustness of DPEC (or NPEC).

**Theorem 5** (Robustness)**.** *Suppose that* $M'$ *is a DPRC or NPRC, then* $M'$ *can take any string of symbols* $A \in \pi^*$ *as an input, i.e., the transition function of a DPRC or NPRC* $M'$ *can be generalized to a function* $\delta' : Q \times \pi^* \to D(Q)$.

**Proof.** The proof is similar to proof of robustness of DPEC and NPEC. □

**Definition 14.** *The language accepted by a DPRC or NPRC $M'$ is defined as a function $L_{M'} : \pi^* \to [0,1]$: for any $x = a_1a_2 \cdots a_n \in \pi^*$,*

$$L_{M'}(x) = \sum_{q\in F, q_i \in Q} \{\delta'(q_0, a_1, q_1) \cdot \delta'(q_1, a_2, q_2) \cdots \delta'(q_{n-1}, a_n, q)\}, i = 1, 2 \cdots n-1.$$

A DPRC $M'$ also inherits the relationship of accepted languages between DPEC and PA.

**Theorem 6** (Semantic generalization)**.** *Suppose that DPRC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ is a semantic generalization of PA $M = (Q, \Sigma, \delta, q_0, F)$ and the languages accepted by them are functions $L_{M'}$ and $L_M$ respectively. The languages accepted by them have the following properties:*

1. *for any $x \in \Sigma^*$, $L_{M'}(x) = L_M(x)$,*
2. *for any $x \in \pi^*$, there exist $x' \in \Sigma^*$ such that $L_{M'}(x) \le L_{M'}(x') = L_M(x')$.*

**Proof.** if $x = \varepsilon$ or $x \in \Sigma^*$, the proof is similar to the proof of semantic generalization of DPEC and NPEC. For any $x = a_1a_2 \cdots a_n \in \pi/\Sigma^*$,

$$L_{M'}(x) = \sum_{q\in F, q_i \in Q} \{\delta'(q_0, a_1, q_1) \cdot \delta'(q_1, a_2, q_2) \cdots \delta'(q_{n-1}, a_n, q)\}, i = 1, 2 \cdots n-1.$$

Since $sim(a, a^*) \le 1$, $\delta'(q, a, p) = \delta(q, a^*, p) \cdot sim(a, a^*) \le \delta(q, a^*, p)$, for any $q, p \in Q$. Let $x = a_1^*a_2^* \cdots a_n^* \in \Sigma^*$.

$$\begin{aligned}
& L_{M'}(x) \\
& = \textstyle\sum_{q\in F, q_i \in Q} \{\delta'(q_0, a_1, q_1) \cdot \delta'(q_1, a_2, q_2) \cdots \delta'(q_{n-1}, a_n, q)\} \\
& \le \textstyle\sum_{q\in F, q_i \in Q} \{\delta(q_0, a_1^*, q_1) \cdot \delta(q_1, a_2^*, q_2) \cdots \delta(q_{n-1}, a_n^*, q)\} \\
& = L_{M'}(x').
\end{aligned}$$

And since $x = a_1^*a_2^* \cdots a_n^* \in \Sigma^*$, by (1), we can get $L_{M'}(x') = L_M(x')$. □

**Theorem 7** (Semantic generalization)**.** *Suppose that DPRC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ and NPRC $M'' = (Q, \Sigma, \delta, \Sigma'', \delta'', q_0, F)$ are semantic generalizations of PA $M = (Q, \Sigma, \delta, q_0, F)$ and the languages accepted by them are functions $L_{M'}$, $L_{M''}$ and $L_M$ respectively. The languages accepted by them have the following properties:*

1. *for any $x \in \Sigma^*$, $L_{M''}(x) = L_{M'}(x) = L_M(x)$,*
2. *for any $x \in \pi^*$, there exist $x' \in \Sigma^*$ such that $L_{M''}(x) \ge L_{M'}(x) \le L_M(x')$.*

**Proof.** if $x = \varepsilon$ or $x \in \Sigma^*$, the proof is similar to the proof of semantic generalization of DPEC and NPEC. For any $x = a_1a_2 \cdots a_n \in \pi/\Sigma^*, p \in Q$,

$$\begin{aligned}
& \delta''(q, a, p) \\
& = min\{\textstyle\sum_{a^*\in\Sigma} \delta(q, a^*, p) \cdot sim(a, a^*), 1\} \\
& \ge \delta(q, a^*, p) \cdot sim(a, a^*) \\
& = \delta'(q, a, p).
\end{aligned}$$

Therefore, $L_{M''}(x) \ge L_{M'}(x')$. By the semantic generalization of DPRC, we can get that there exist $x' \in \Sigma^*$ such that $L_{M'}(x) \le L_M(x')$. □

The above two theorems show the relationship between the language accepted by NPRC or DPRC $M'$ and the corresponding PA $M$. Every element of $M$ is included in $M'$ and the languages accepted by $M$ can be taken as a part of the languages accepted by $M'$. Therefore, $M'$ is a semantic generalization of $M$.

**Example 4** (Bookshop)**.** *Let $M = (Q, \Sigma, \delta, q_0, F)$ be a PA. Suppose that $M$ is applied to an application of recommendation system for a book shop. When a customer inputs the name of a book and buys a book, PA $M$ recommends a list of books, ordered by the probability of which book the customer will buy as well. Suppose that $Q$ is the set of books, i.e.,*

$$Q = \{q_1 = \text{Web Ontology Languages}, q_2 = \text{Knowledge Representation}, q_3 = \text{Artificial Intelligence}, \\ q_4 = \text{Theory of Computation}\}.$$

*$\Sigma$ includes all combinations of the key words of every book's name in $Q$ which means* $\Sigma = \{$Web, Ontology, Languages, Web Ontology, Web Ontology Languages, Knowledge, Representation, Knowledge Representation$\cdots\}$. *$\delta(q, a, p)$ means the probability of recommending book $p$ to a customer after this customer has bought $q$ with input $a$. The value of every $\delta(q, a, p)$ can be calculated by historical data which means recommendation system is based on customer's purchasing history.*

*But as a user, when I have to input "Web Ontology Languages", I prefer to input "WOL"—short for "Web Ontology Languages"—because it is more efficient to input an acronym. When I have to input "Knowledge Representation" or "Artificial Intelligence", I also prefer to input "KR" or "AI" as well.*

*On the other hand, as a designer, the inputs transmitted from users are unpredictable. Users may input acronyms, synonyms or equivalent concepts in semantics, even some strange symbols coming from spelling mistakes. Therefore, when defining the input alphabet $\Sigma$, we need to collect synonyms or equivalent concepts of every element of $\Sigma$, as many as we can. But when $\Sigma$ is very big, it is too hard to collect all the synonyms or equivalent concepts. Another worse case is that, as the development of globalization, the users may come from any country and input symbols with any language (English, Chinese, French, and so on). For robustness and universality, as a designer, it is too hard to define a PA $M$ with the ability to handle all those cases, because PA is restricted by a fixed finite input alphabet $\Sigma$ defined previously. But a NPRC is competent for this challenge.*

*Hence, we rebuild PA $M$ as a NPRC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ as follows. Firstly, define* $\Sigma' = \{$linguistics, represent$\cdots\}$ *as the set of some possible inputs. For every $a \in \Sigma'$, define $\delta'(q, a, p) = \delta(q, a, p) \cdot sim(a, a^*)$. Notice that after we have defined the way to compute semantic similarity, we only need to collect familiar equivalents and synonyms as $\Sigma'$. When an input $a \notin \Sigma \cup \Sigma'$ transmitted from customers, we add $a$ to $\Sigma'$ and find $a^*$ based on function sim. Then define $\delta'(q, a, p) = \delta(q, a^*, p) \cdot sim(a, a^*), q, p \in Q$.*

*For instance, we consider a sample case. In PA $M = (Q, \Sigma, \delta, q_0, F)$,* $Q = \{q =$ Web Ontology Languages, $p$ = Knowledge Representation$\}$. $\Sigma = \{$Web, Ontology, Languages, Representation, Knowledge$\}$. *$q_0 = q, F = Q$. $\delta$ is defined as:*

$$\begin{aligned}
&\delta(q, Web, q) = 0.7, \delta(q, Ontology, q) = 0.8, \delta(q, Languages, q) = 0.9, \\
&\delta(q, Representation, q) = 0.1, \delta(q, Knowledge, q) = 0.2, \\
&\delta(q, Web, p) = 0.3, \delta(q, Ontology, p) = 0.2, \delta(q, Languages, p) = 0.1, \\
&\delta(q, Representation, p) = 0.9, \delta(q, Knowledge, p) = 0.8, \\
&\delta(p, Web, q) = 0.1, \delta(p, Ontology, q) = 0.4, \delta(p, Languages, q) = 0.5, \\
&\delta(p, Representation, q) = 0.7, \delta(p, Knowledge, q) = 0.9, \\
&\delta(p, Web, p) = 0.9, \delta(p, Ontology, p) = 0.6, \delta(p, Languages, p) = 0.5, \\
&\delta(p, Representation, p) = 0.3, \delta(p, Knowledge, p) = 0.1.
\end{aligned}$$

$\delta(q, Web, p) = 0.3$ *means after a customer inputs key words* "Web" *searching for a book and finally buys book* "Web Ontology Languages", *the probability of he will buy book* "Knowledge Representation" *is 0.3. Others are similar.*

*In order to deal with similar inputs, we rebuild PA $M$ to a NPRC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ as follows. Suppose that we only consider the additional case that customers will input* "linguistics" *and* "represent" *searching for a book, that is, $\Sigma' = \{linguistics, represent\}$ and $\delta'(k, a, k') = \delta(k, a, k')$, for $k, k' \in Q, a \in \Sigma$. Suppose*

$$sim(linguistics, Languages) = 0.9, sim(represent, Representation) = 0.7$$

*and for other case* $a' \in \Sigma', a \in \Sigma, sim(a, a') = 0$. *For* $a \in \Sigma'$, $\delta'$ *is defined as:*

$$\begin{aligned}
&\delta'(q, linguistics, p) = \delta(q, Languages, p) \cdot sim(linguistics, Languages) = 0.9 * 0.1 = 0.09,\\
&\delta'(q, represent, p) = \delta(q, Representation, p) \cdot sim(represent, Representation) = 0.7 * 0.9 = 0.63,\\
&\delta'(q, linguistics, q) = \delta(q, Languages, q) \cdots sim(linguistics, Languages) = 0.9 * 0.9 = 0.81,\\
&\delta'(q, represent, q) = \delta(q, Representation, q) \cdot sim(represent, Representation) = 0.7 * 0.1 = 0.07,\\
&\delta'(p, linguistics, q) = \delta(p, Languages, q) \cdot sim(linguistics, Languages) = 0.9 * 0.5 = 0.45,\\
&\delta'(p, represent, q) = \delta(p, Representation, q) \cdot sim(represent, Representation) = 0.7 * 0.7 = 0.47,\\
&\delta'(p, linguistics, p) = \delta(p, Languages, p) \cdot sim(linguistics, Languages) = 0.9 * 0.5 = 0.45,\\
&\delta'(p, represent, p) = \delta(p, Representation, p) \cdot sim(represent, Representation) = 0.7 * 0.3 = 0.21.
\end{aligned}$$

*With the above extended* $\Sigma'$ *and* $\delta'$*, when an input "linguistics" that is not in* $\Sigma$ *transmitted to automaton, PA M is invalid, but NPRC is valid.*

An NPRC can be modified as a traditional probabilistic automaton as follows.

**Definition 15** (NPRC)**.** *A nondeterministic probabilistic automaton for semantic computing under related concept (NPRC) is a seven-tuple* $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$*, where*

- $M = (Q, \Sigma, \delta, q_0, F)$ *is a PA,*
- $\Sigma' = \{a | a \in \pi, \exists a^* \in \Sigma\}$,
- $\delta' : Q \times (\Sigma \cup \Sigma') \to D(Q)$ *is a generalized transition function of* $\delta$*:*

$$\delta'(q, a) = \begin{cases} \delta(q, a), \ \ if \ \ a \in \Sigma, \\ \frac{1}{\sum_{a^* \in \Sigma} sim(a, a^*)} \sum_{a^* \in \Sigma} \delta(q, a^*) \cdot sim(a, a^*), \ \ if \ \ a \in \Sigma'. \end{cases}$$

The proof of $\sum_{p \in Q} \delta'(q, a)(p) = 1$ is similar to NPEC. The key to proof this is that for different most similar concepts $a^*, a^{**}$ of $a$,

$$sim(a, a^*) = sim(a, a^{**}) = max_{a' \in \Sigma}\{sim(a, a')\}.$$

After the above modification of transition function, NPRC becomes a traditional probabilistic automaton. It is obvious that the property of robustness is still valid. But recall that the main purpose of this paper is to build computational models for semantic computing. Therefore, we prefer to modify it to the following forms.

**Definition 16** (NPRC)**.** *A nondeterministic probabilistic automaton for semantic computing under related concept (NPRC) is a seven-tuple* $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$*, where*

- $M = (Q, \Sigma, \delta, q_0, F)$ *is a PA,*
- $\Sigma' = \{a | a \in \pi, \exists a^* \in \Sigma\}$, $N(a^*)$ *is number of* $a^*$ *corresponding to a,*
- $\delta' : Q \times (\Sigma \cup \Sigma') \to D(Q)$ *is a generalized transition function of* $\delta$*:*

$$\delta'(q, a) = \begin{cases} \delta(q, a), \ \ if \ \ a \in \Sigma, \\ \frac{1}{N(a^*)} \sum_{a^* \in \Sigma} \delta(q, a^*) \cdot sim(a, a^*), \ \ if \ \ a \in \Sigma'. \end{cases}$$

Based on the above form of NPRC, for any DPRC and NPRC, they have the following property.

**Theorem 8** (Semantic computing)**.** *For any DPRC or NPRC* $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$*, for any* $a \in \Sigma'$, $\sum_{p \in Q} \delta'(q, a)(p) = sim(a, a^*)$.

**Proof.** For any DPRC $M'$ and any $a \in \Sigma'$,

$$\begin{aligned}
&\textstyle\sum_{p\in Q}\delta'(q,a)(p)\\
&= \textstyle\sum_{p\in Q}\delta(q,a^*)(p)\cdot sim(a,a^*)\\
&= sim(a,a^*)\textstyle\sum_{p\in Q}\delta(q,a^*)(p)\\
&= sim(a,a^*) * 1\\
&= sim(a,a^*).
\end{aligned}$$

For any NPRC $M'$ and any $a \in \Sigma'$,

$$\begin{aligned}
&\textstyle\sum_{p\in Q}\delta'(q,a)(p)\\
&= \textstyle\sum_{p\in Q}\frac{1}{N(a^*)}\sum_{a^*\in\Sigma}\delta(q,a^*)(p)\cdot sim(a,a^*)\\
&= \frac{1}{N(a^*)}\textstyle\sum_{a^*\in\Sigma} sim(a,a^*)\sum_{p\in Q}\delta(q,a^*)(p)\\
&= \frac{1}{N(a^*)}\textstyle\sum_{a^*\in\Sigma} sim(a,a^*)\\
&= \frac{1}{N(a^*)} * N(a^*) * sim(a,a^*)\\
&= sim(a,a^*).
\end{aligned}$$

□

Therefore, in a DPRC or NPRC $M'$, for any $a \in \Sigma'$, $\sum_{p\in Q}\delta'(q,a)(p)$ turn out to be the semantic similarity of two concepts: $sim(a,a^*)$. It means the total probability of that a DPRC or NPRC will enter all the states with a similar input is the semantic similarity between generalized input and similar input.

In this subsection, we have defined different kinds of NPRC, because in different cases, we need different transition function. For an NPRC $M'$ based on Definition 13, any generalized input is related to all similar inputs. But a NPRC $M'$ based on Definition 15 is a traditional probabilistic automaton which satisfies the properties of traditional probabilistic automaton. For a NPRC $M'$ based on Definition 16, it is a probabilistic automaton model for semantic computing.

In the above subsections, traditional probabilistic automata are generalized for semantic computing under equivalent and related concepts or words, respectively. Compared with traditional probabilistic automata, generalized probabilistic automata are more robust, because it is still valid when the input transmitted from users is not in the defined input alphabet $\Sigma$. It explains the semantic computing from computation theory. In the following section, we will show the robustness with an application for the weather forecast.

## 4. Application

The theory of computation has been applied to different areas such as social networks [23], clustering [24], and e-Services [25]. In this paper, we will build a machine for weather forecast [26–29] based on automata.

For a given location and time, the primary mission of weather forecast is to predict the conditions of the atmosphere. Human beings have been attempted to predict the weather informally for millennia, because we need weather warnings to protect life and property. Forecasts based on temperature and precipitation are essential to agriculture. We need to be prepared for floods and droughts in advance anyway. In our daily life, the weather forecast is an important factor for the plan of outdoor activities.

The understanding of atmospheric physics builds the foundation of modern numerical weather prediction [30]. After collecting quantitative data about the current state of the atmosphere at a given place, weather forecasts are made by using meteorology to predict how the atmosphere will change. The basic idea of numerical weather prediction is that: firstly, sample the state of the fluid at a given location and time. Then use the equations of fluid dynamics and thermodynamics to estimate the state of the fluid at some time in the future. In Reference [28], they propose that:

*The weather station variables analyzed were air temperature, RH, 6-m (20 ft) wind speed, precipitation, cloud cover and solar radiation. Note that in keeping with standard fire weather terminology we will refer to the 6-m wind speed as the 20-ft wind speed throughout the rest of the paper."*

Therefore, in this section, we define a factor that impacts weather as a six-tuple

$$a = (\text{air temperature,RH, 6-m wind speed, precipitation, cloud cover, solar radiation}).$$

A numerical representation of the above six-tuple is called a weather factor, which means that the change of weather state is based on the changing trend of—temperature, RH, 6-m wind speed, precipitation, cloud cover and solar radiation. We want to build a probabilistic automaton PA $M = (Q, \Sigma, \delta, q_0, F)$ for weather forecast based on finite historical observation data. The intuitive idea is that $Q$ is the set of weather states: sunny, cloudy, rainy, and so on. $\Sigma$ is the set of weather factors. $\delta(q, a)(p)$ represents the probability of weather state changing to $p$ from $q$ with weather factor $a$.

The challenge is that the input alphabet of traditional probabilistic automata is a fixed finite set. But in the natural environment, the weather has been changing all the time. It is impossible and impractical to define every case of weather factors. We will analyze the problem of lacking data in Table 1 lately. One compromised solution is that we extract a finite set of weather factors from historical observation data, defined as $\Sigma$. When it comes to a new weather factor $a \notin \Sigma$, we find a most similar weather factor $a^* \in \Sigma$ such that $sim(a, a^*) = max_{a' \in \Sigma}\{sim(a, a')\}$. Then we take $a$ as $a^*$. With this idea, we rebuild PA $M$ as a DPRC $M' = (Q, \Sigma, \Sigma', \delta, \delta', q_0, F)$ based on historical observation data for weather forecast.

**Table 1.** The approximate probability of a new observed weather factor $a \in \Sigma$ in different cases.

| Interval (Year) | Frequency (1/min) | Amount | Probability (Approximate) |
|---|---|---|---|
| 10 | 10 | $5.3 \times 10^5$ | 0 |
| 10 | 1 | $5.3 \times 10^6$ | 0 |
| 50 | 10 | $2.6 \times 10^6$ | 0 |
| 50 | 1 | $2.6 \times 10^7$ | 0 |
| 100 | 10 | $5.3 \times 10^6$ | 0 |
| 100 | 1 | $5.3 \times 10^7$ | 0 |

Firstly, we build a PA $M = (Q, \Sigma, \delta, q_0, F)$. For convenience, the six factors of temperature, RH, 6-m wind speed, precipitation, cloud cover and solar radiation are normalized to unit interval $I = [0, 1]$ by normalize function:

$$f(x) = \frac{x - min}{max - min},$$

where $min$ and $max$ represent the minimum and maximum value that take into account. For instance, if we only consider the weather temperature between $-20$ degree centigrade and 50 degree centigrade, then every temperature $t$ is normalized by:

$$f(t) = \frac{t + 20}{70}.$$

Suppose that $\Sigma$ is a finite set of weather factors extracted from historical observation data. Because of normalization, every $a \in \Sigma$ is a tuple $a = (a_1, a_2 \cdots a_6), |a_i| \in I$ where $a_i$ corresponding to the change of six factors: temperature, RH, 6-m wind speed, precipitation, cloud cover and solar radiation respectively. The sign of $a_i$ represents the corresponding factor is increasing (+) or decreasing ($-$) and the absolute value $|a_i|$ describe the rate of change. For instance, if $a_1 = -0.2$, then it means the temperature is decreasing and its rate is 0.2. $a_2 = 0.3$ means RH is increasing and its rate is 0.3. In fact, $a \in \Sigma$ is the changing trend of weather states. $\delta$ is defined as $\delta(q, a)(p) = \frac{n(q,a,p)}{|\Sigma|}$ where $|\Sigma|$

is the total of all weather factors in $\Sigma$ and $n(q,a,p)$ is the number of times that the weather in state $q$ changes to state $p$ with weather factor $a$ in historical observation data. It means the probability of weather in state $q$ change to state $p$ with weather factor $a$ is $n(q,a,p)/|\Sigma|$. Let $F = Q$ and $q_0$ be the first observation record. Then PA $M$ can show the probability of weather states changing from a state to another state based on historical observation data.

Suppose that we have been recorded weather factors for $\Sigma$ every 10 min for 10 years. Roughly, the total number of records in $\Sigma$ is $10 * 365 * 24 * 6 = 525{,}600 \approx 5.3 \times 10^5$. However, if the six elements in a weather factor $a \in \Sigma$ is accurate to 0.01, every element will has 201 cases between $-1.00$ to 1.00. Roughly, the number of total possible cases of weather factors is $(200)^6 \approx 6.4 \times 10^{13}$. Hence, for a random observed weather factor $a$, the probability of $a \in \Sigma$ is:

$$P(a \in \Sigma) = \frac{5.3 \times 10^5}{6.4 \times 10^{13}} \approx 0.$$

Table 1 show the approximate probability of $a \in \Sigma$ in different cases. For example, the data in fist line means that if we record weather factors every 10 minutes for 10 years, we can get approximate $5.3 \times 10^5$ records and the probability of a new observed weather factor $a \in \Sigma$ is near 0.

Table 1 shows that even when we collect data from records over 100 years, it is still not enough to build $\Sigma$ for PA $M$. Therefore PA $M$ is not competent for the weather forecast. Because when we observe and get the current weather factor $a$, the probability of $a \in \Sigma$ is almost 0. The weather has been changing all the time. There are so many possible cases of weather factors, even though we have been recording every minute for 100 years, it is still far from enough. It seems impossible to make sure that every time the observed weather factor $a$ will be in the set $\Sigma$. Once the current weather factor $a$ is not in $\Sigma$, PA $M$ cannot work.

In order to build an automaton for weather forecast, we need to redefine PA $M$ as a DPRC $M' = (Q, \Sigma, \delta, \Sigma', \delta', q_0, F)$ where $\Sigma'$ is the set of all possible weather factors, that is, $\Sigma' = E^6, E = [-1, 1]$. Recall that as the robustness of DPRC, $\delta'$ can be generalized to a function $\delta' : Q \times \pi^* \to D(Q)$. In this application, $\delta'$ is generalized to $\delta' : Q \times (E^6)^* \to D(Q)$.

The semantic similarity of two weather factors $a, b \in \Sigma$ is defined as $sim : E^6 \times E^6 \to E$ and here we use the most classical cosine function to compute similarity, that is,

$$sim(a,b) = cos(a,b) = \frac{a \cdot b}{|a| \cdot |b|} = \frac{\sum\limits_{i=1}^{6} a_i b_i}{\sum\limits_{i=1}^{6} a_i^2 \sum\limits_{i=1}^{6} b_i^2},$$

where $a = (a_1, a_2 \cdots a_6)$ and $b = (b_1, b_2 \cdots b_6)$. Transition function $\delta'$ is defined as

$$\delta'(q,a)(p) = \begin{cases} \delta(q,a)(p), & for\ \ a \in \Sigma, \\ \delta(q,a)(p) \cdot sim(a,a^*), & for\ \ a \in E^6/\Sigma, \end{cases}$$

where $a^*$ satisfy $sim(a,a^*) = max_{a' \in \Sigma}\{sim(a,a')\}$.

The intuitive idea is to suppose the current weather factor is $a$ and the weather state is $p$. When we want to know the probability of current weather $q$ changing to state $p$ with weather factor $a$, we go back to check the records. By semantic similarity function $sim(a,b)$ we find that $a^* \in \Sigma$ is the most similar weather factor to $a$. The probability of $q$ changing to $p$ with weather factor $a$ can be obtained from PA $M$ by $\delta(q,a^*)(p)$. Then we can conclude that the probability of current state $q$ changing to state $p$ is $\delta(q,a^*)(p) \cdot sim(a,a^*)$. Then, by DPRC $M'$, we can get the probability that the current weather state may change to any weather states with any weather factor.

Table 2 show the comparison of the data that needed for PA $M$ and DPRC $M'$. Suppose that we only have the data recorded every minute in last 10 years. Table 1 show the comparison of the data that needed for PA $M$ and DPRC $M'$. PA $M$ needs $6.4 \times 10^{13}$ to build $\Sigma$. The data we have is far more enough and it is impossible to get so many data. If we must get it, the cost will be out of control. But DPRC $M'$ dose not has this problem. The data in the last 10 years is enough.

**Table 2.** Comparison of the data that needed for PA $M$ and DPRC $M'$.

| Automaton | How Many Data We Have | If the Data Is Enough | How Many Data Are Needed | Cost |
|---|---|---|---|---|
| PA $M$ | $5.3 \times 10^6$ | no | $6.4 \times 10^{13}$ | out of control |
| DPRC $M'$ | $5.3 \times 10^6$ | yes | $5.3 \times 10^6$ | under control |

Another challenge is the precision of implements used to measure the weather factors. DPRC $M'$ can only deal with precise information. Suppose that when measuring a weather factor, the result obtained from implements is $a$, but because of the precision of the implements, in fact the real weather factor may be $a + k_a$ ($k_a$ is deviation of $a$) as well. In this case, the weather factor is imprecise information which cannot be dealt with by DPRC $M'$ but it can be described by distribution. Suppose $W_a = 0.9/a + 0.1/(a + k_a)$ means that when the result got from implements is $a$, the real weather factor may be $a$ with the probability 0.9 and may be $a + k_a$ with the probability 0.1 caused by the precision of implements. In this case, we need to generalize above DPRC $M'$ as a model $M'' = (Q, \Sigma_w, \delta, \Sigma'_w, \delta'', q_0, F)$ where $\Sigma'_w = \{W_a = 0.9/a + 0.1/(a + k_a) | a \in \Sigma'\}$. $\delta'' : Q \times \Sigma'_w \to D(Q)$ is defined as:

$$\begin{aligned}
&\delta''(q, W)(p) \\
&= \textstyle\sum_{a \in \Sigma} \delta'(q, a)(p) W(a) \\
&= \textstyle\sum_{a \in \Sigma} \delta'(q, a)(p) W_a(a) \\
&= 0.9 * \delta'(q, a)(p) + 0.1 * \delta'(q, a_k)(p) \\
&= 0.9 * \delta(q, a)(p) * sim(a, a^*) + 0.1 * \delta(q, a_k)(p) * sim(a, a_k^*).
\end{aligned}$$

Therefore, even though PA $M$ is incompetent to weather forecast, but DPRC $M'$ is competent. The critical difference is that PA $M$ is restricted in a fixed finite input alphabet $\Sigma$, but DPRC $M'$ can take any weather factor as input because of robustness. Even the input is not in the fixed input alphabet $\Sigma$ defined previously, DPRC can take it as a similar case and give a similar result based on semantic similarity.

## 5. Conclusions

Traditional automata will be invalid when an undefined input is transmitted from users in applications. In this paper, we generalize probabilistic automata for semantic computing. The generalized automata are more robust, which can take any input as legal input. The semantic generalization of our new probabilistic automata can be reflected from the languages accepted by them, i.e., the languages accepted by a traditional probabilistic automaton are a part of the languages accepted by the corresponding probabilistic automaton under semantic similarity. As shown in the application of weather forecast, when a new weather factor is observed, traditional probabilistic automata become invalid, because the new weather factor hardly belongs to the input alphabet. For robustness, we rebuild traditional probabilistic automata based on semantic similarity. Then when a probabilistic automaton gets an undefined input transmitted from users, despite it is not in the original input alphabet, the automaton is still valid with this undefined input.

Furthermore, probabilistic automata under semantic similarity provide a different understanding of "semantic computing" from a computation theory perspective. In most traditional semantic computing models, computational implementations of semantic reasoning are based on ontology reasoning, rule reasoning, semantic query, and semantic search, and so on. But they are not from the perspective of the formal theory of computation. The semantic computing property of our new probabilistic automata bridges the gap between semantic computing and computation theory. They provide a solid foundation for semantic computing. In our future work, we are planning to generalize the formal models for semantic computing with timed automaton. We are also trying to apply the formal models for semantic computing to deal with the problems such as natural language processing and semantic search.

**Author Contributions:** Conceptualization, G.H.; methodology, G.H.; software, G.H.; validation, G.H., S.H.W. and L.W.; formal analysis, G.H.; investigation, G.H.; resources, G.H.; writing—original draft preparation, G.H. and L.W.; writing—review and editing, G.H. and S.H.W.; visualization, G.H. and S.H.W.; supervision, Y.J.; funding acquisition, Y.J.

**Funding:** The work described in this paper is supported by The National Natural Science Foundation of China under Grant Nos. 61772210 and 61272066; Guangdong Province Universities Pearl River Scholar Funded Scheme (2018); The Project of Science and Technology in Guangzhou in China under Grant No. 201807010043; The key project in universities in Guangdong Province of China under Grant No. 2016KZDXM024.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Liu, H.; Bao, H.; Xu, D. Concept vector for semantic similarity and relatedness based on wordnet structure. *J. Syst. Softw.* **2012**, *85*, 370–381. [CrossRef]
2. Jiang, Y.; Zhang, X.; Tang, Y.; Nie, R. Feature-based approaches to semantic similarity assessment of concepts using Wikipedia. *Inf. Process. Manag.* **2015**, *51*, 215–234. [CrossRef]
3. Jiang, Y.; Bai, W.; Zhang, X.; Hu, J. Wikipedia-based information content and semantic similarity computation. *Inf. Process. Manag.* **2017**, *53*, 248–265. [CrossRef]
4. Baader, F.; Calvanese, D.; McGuinness, D.; Patel-Schneider, P.; Nardi, D. *The Description Logic Handbook: Theory, Implementation and Applications*; Cambridge University Press: Cambridge, UK, 2003.
5. Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; Tompits, H. Combining answer set programming with description logics for the semantic web. *Artif. Intell.* **2008**, *172*, 1495–1539. [CrossRef]
6. Horrocks, I.; Patel-Schneider, P.F.; Bechhofer, S.; Tsarkov, D. OWL rules: A proposal and prototype implementation. *Web Semant. Sci. Serv. Agents World Wide Web* **2005**, *3*, 23–40. [CrossRef]
7. Storey, V.C.; Burton-Jones, A.; Sugumaran, V.; Purao, S. CONQUER: A methodology for context-aware query processing on the World Wide Web. *Inf. Syst. Res.* **2008**, *19*, 3–25. [CrossRef]
8. Sohn, M.; Jeong, S.; Kim, J.; Lee, H.J. Crowdsourced healthcare knowledge creation using patients' health experience-ontologies. *Soft Comput.* **2017**, *21*, 5207–5221. [CrossRef]
9. Tappolet, J.; Kiefer, C.; Bernstein, A. Semantic web enabled software analysis. *Web Semant. Sci. Serv. Agents World Wide Web* **2010**, *8*, 225–240. [CrossRef]
10. Wang, F.; Hu, L.; Zhou, J.; Hu, J.; Zhao, K. A semantics-based approach to multi-source heterogeneous information fusion in the internet of things. *Soft Comput.* **2017**, *21*, 2005–2013. [CrossRef]
11. Jiang, Y. A formal model of semantic computing. *Soft Comput.* **2018**, 1–19. [CrossRef]
12. Aouicha, M.B.; Taieb, M.A.H.; Hamadou, A.B. SISR: System for integrating semantic relatedness and similarity measures. *Soft Comput.* **2018**, *22*, 1855–1879. [CrossRef]
13. Li, Y.; Bandar, Z.A.; McLean, D. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. Knowl. Data Eng.* **2003**, *15*, 871–882.
14. Pirró, G. A semantic similarity metric combining features and intrinsic information content. *Data Knowl. Eng.* **2009**, *68*, 1289–1308. [CrossRef]
15. Qu, R.; Fang, Y.; Bai, W.; Jiang, Y. Computing semantic similarity based on novel models of semanticrepresentation using Wikipedia. *Inf. Process. Manag.* **2018**, *54*, 1002–1021. [CrossRef]
16. Cao, Y.; Xia, L.; Ying, M. Probabilistic automata for computing with words. *J. Comput. Syst. Sci.* **2013**, *79*, 152–172. [CrossRef]
17. Qiu, D.; Wang, H. A probabilistic model of computing with words. *J. Comput. Syst. Sci.* **2005**, *70*, 176–200. [CrossRef]
18. Shamsizadeh, M.; Zahedi, M. Intuitionistic general fuzzy automata. *Soft Comput.* **2016**, *20*, 3505–3519. [CrossRef]
19. Sipser, M. *Introduction to the Theory of Computation*; Thomson Course Technology: Boston, MA, USA, 2006; Volume 2.
20. Ying, M. A formal model of computing with words. *IEEE Trans. Fuzzy Syst.* **2002**, *10*, 640–652. [CrossRef]

21. Zadeh, L.A. Fuzzy Languages and Their Relation to Human and Machine Intelligence. In *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*; World Scientific: Singapore, 1996; pp. 148–179.
22. Couto, F.M.; Silva, M.J.; Coutinho, P.M. Measuring semantic similarity between Gene Ontology terms. *Data Knowl. Eng.* **2007**, *61*, 137–152. [CrossRef]
23. Moradabadi, B.; Meybodi, M.R. Link prediction in weighted social networks using learning automata. *Eng. Appl. Artif. Intell.* **2018**, *70*, 16–24. [CrossRef]
24. Mohammad, H.M.; Alireza, R. Learning Automata Clustering. *J. Comput. Sci.* **2018**, *24*, 379–388.
25. El-Qurna, J.; Yahyaoui, H.; Almulla, M. A new framework for the verification of service trust behaviors. *Knowl. Based Syst.* **2017**, *121*, 7–22. [CrossRef]
26. Sobash, R.A.; Schwartz, C.S.; Romine, G.S.; Fossell, K.R.; Weisman, M.L. Severe weather prediction using storm surrogates from an ensemble forecasting system. *Weather Forecast.* **2016**, *31*, 255–271. [CrossRef]
27. Halperin, D.J.; Torn, R.D. Diagnosing conditions associated with large intensity forecast errors in the Hurricane Weather Research and Forecasting (HWRF) model. *Weather Forecast.* **2018**, *33*, 239–266. [CrossRef]
28. Page, W.G.; Wagenbrenner, N.S.; Butler, B.W.; Forthofer, J.M.; Gibson, C. An Evaluation of NDFD Weather Forecasts for Wildland Fire Behavior Prediction. *Weather Forecast.* **2018**, *33*, 301–315. [CrossRef]
29. Guan, H.; Zhu, Y. Development of verification methodology for extreme weather forecasts. *Weather Forecast.* **2017**, *32*, 479–491. [CrossRef]
30. Richardson, L.F. *Weather Prediction by Numerical Process*; Cambridge University Press: Cambridge, UK, 2007.